Санкт-Петербургский Государственный Университет
факультет Прикладной математики - процессов управления

Выпускная квалификационная работа

Методы хранения и обработки больших объемов пространственных данных в растровых форматах

Выполнил: аспирант
3 года обучения
Соловьев П. А.

Научный руководитель:
Сергеев С. Л.,
к.ф.-м.н., доцент

Рецензент:
Афонин А. Н.
к.с.-х.н., доцент

Санкт-Петербург
2016г.

Реферат

В данной работе содержится: 48 страниц текста, 7 разделов; в данной работе используются ссылки на 49 источников. В работе содержится 3 таблицы, 8 формул, 12 рисунков. К работе прилагаются 4 приложения.
Работа написана на русском языке; к работе прилагаются аннотации на русском и английском языках.

Ключевые слова: Большие данные, распределенные вычислительные системы, добровольные вычислительные сети, Географические информационные системы, ГИС, эколого-географический анализ.

Аннотация

В работе рассматривается вопрос о возможности использования распределенных географических информационных систем, работающих в добровольных и гибридных вычислительных сетях, для решения задач по хранению и анализу больших объемов пространственных данных. В работе предлагается концепция распределенной ГИС и производится тестирование ее прототипа. По полученным результатам тестирования делается вывод о пригодности распределенных ГИС, работающих в добровольных и гибридных вычислительных сетях, для хранения пространственных «больших данных» в растровых форматах и для проведения эколого-географического анализа этих данных.

Abstract

The aim of this work is to study the possibility of using distributed geographic information systems for volunteer and hybrid networks to store and process and analyze spatial bigdata in raster formats. A concept of a distributed GIS is discussed and its prototype is tested. Based on the results of this testing a conclusion about the suitability of distributed geographic information systems for volunteer and hybrid networks for storing and processing spatial bigdata in raster formats is drawn.

Оглавление

1. Введение

В современном мире человек и создаваемые им информационные системы все чаще сталкиваются с необходимостью хранить, обрабатывать и перемещать колоссальные объемы данных. Это является верным для самых различных областей человеческой деятельности, и эколого-географический анализ не является исключением.

Так, существующие в настоящее время технологии дистанционного зондирования дают возможность в короткие сроки и без значительных финансовых затрат получать высокоточные снимки земной поверхности в самых различных диапазонах электромагнитного спектра. Одновременно с этим, развитие и совершенствование вычислительной техники позволяет быстро и детально анализировать собираемые данные. Наконец, благодаря повсеместной доступности сети Internet в современном мире, заинтересованные специалисты, находящиеся в любых точках земного шара, имеют моментальный доступ как к исходным пространственным данным, так и ко всевозможным результатам их обработки.

Это приводит к тому, что объемы пространственных данных, в первую очередь представленных растровыми форматами, растут едва ли не геометрически. А так как пространственные данные в настоящее время активно используются в самых различных областях научной и повседневной деятельности человека, естественным образом возникает необходимость в надежных и производительных вычислительных системах для хранения, обработки и передачи пространственных данных.

Актуальность разработки географических информационных систем, ориентированных на работу с «большими данными», в современном мире сложно переоценить: использование подобных систем позволяет прогнозировать и отслеживать развитие различного рода чрезвычайных ситуаций; проводить виртуальные биологические, экологические и геологические эксперименты направленные на расширение наших знаний о прошлом и настоящем мира в котором мы живем; а также собирать информацию для более качественного использования земельных и биологических ресурсов при одновременном уменьшении негативного влияния человеческой активности на окружающую среду [1].

2. Определение термина «большие данные»

Несмотря на частоту с которой данный термин употребляется в обсуждениях современных компьютерных технологий, у него отсутствует единое общепризнанное определение. Большинство используемых определений термина «большие данные» можно отнести к одному из трех основных классов [2].

Атрибутивные определения Данный класс определений берет свое начало от «4Vs»-определения, предложенного в 2011 г. работниками International Data Corporation (IDC) [3], одного из пионеров и лидеров в области исследований «больших данных» и их влияния на современные компьютерные технологии. Чтобы являться «большими данным» по атрибутивному определению, данные должны обладать четырьмя ключевыми свойствами:

- **Объем** (Volume) – «Большие данные» есть генерируемые или хранимые данные таких объемов, с которыми стандартное ПО не может справиться за удовлетворительное время [4]. Объем является основным критерием, в соответствие с которым те или иные данные относят к «большим данным».
- **Разнообразие** (Variety) – «Большие данные» есть данные со значительным разнообразием в своих источниках и своей природе.
- **Скорость накопления** (Velocity) – «Большие данные» есть такие данные, объемы которых растут с чрезвычайно-большими скоростями.
- **Достоверность** (Veracity) – «Большие данные» есть данные, существенная часть которых может быть неточной или недостоверной.

Атрибутивные определения являются развитием «3Vs»-определения, которое было предложено еще в 2001 г. аналитиком META Group Inc. Дугласом Лэней (Douglas Laney) [5]. Согласно этому определению, «большие данные» обладают следующими свойствами: объем, разнообразие и скорость накопления.

«3Vs»-определение в настоящее время считается устаревшим, однако, им по-прежнему пользуются некоторые из специалистов IBM [6] и Microsoft [7].

Сравнительные определения Определения данного класса описывают «большие данные» как такие наборы данных, которые из-за своего размера не могут быть сохранены или обработаны при помощи типичных Систем Управления Базами Данных [8, 9, 10]. Сравнительные определения являются субъективными и не определяют «большие данные» в терминах какой-либо конкретной метрики. С другой стороны, определения данного класса включают эволюционный аспект в описание тех наборов данных, которые могут быть причислены к «большим данным».

Архитектурные определения Данный класс определений берет свое начало из работы [11] сотрудников Национального Института Стандартов и Технологий США (NIST), опубликованной в 2012 г. С точки зрения архитектурных определений, «большими данными» будут считаться такие данные, объемы, скорость накопления или особенности обработки которых ограничивают возможность эффективного анализа с использованием традиционных реляционных подходов или требуют значительного горизонтального масштабирования БД для своей эффективной обработки.

Архитектурный подход к определению «больших данных» позволяет выделить две подобласти: «Big Data Science», т.е. исследование техник по получению и оцениванию «больших данных» и «Big Data Frameworks», т.е. библиотеки ПО и сопутствующие им алгоритмы для выполнения распределенной обработки и анализа «больших данных» на вычислительных кластерах.

Поскольку разница между этими подходами к определению «больших данных» заключается в основном в том, на каком из указанных аспектов «больших данных» концентрируется внимание, возможно объединить их в единое целое. Такой подход позволяет составить следующую таблицу для сравнения традиционных данных с «большими данными» [2].

	традиционные данные	**«большие данные»**
Объем	Gb	постоянно увеличивается (Tb или Pb в настоящее время)
Скорость накопления	за час, день...	более быстрое
Структурированность	структурированные данные	слабоструктурированные или неструктурированные данные
Источник данных	централизованный	полностью распределенный
Сложность интеграции	низкая	высокая
Хранение данных	Реляционные БД	HDFS, NoSQL БД
Доступ к данным	интерактивный	пакетный, в режиме реального времени

Таблица 1: Сравнение традиционных данных и «Больших данных»

Как можно видеть из таблицы 1, «большие данные» отличаются от традиционных данных большим объемом, причем нижняя граница того, что же может считаться «большими данными» постоянно повышается. В настоящее время «большие данные» начинаются как минимум от нескольких терабайт, а как максимум – от нескольких петабайт. Причем эти данные генерируются с очень большой скоростью: например, за одну секунду своей работы Большой Адронный Коллайдер, в зависимости от того, какие из детекторов задействованы в ходе эксперимента, производит от 600 Mb до 25 Gb данных [12].

Помимо своих больших объемов и высокой скорости генерации и/или накопления, «большие данные» также производятся большим числом источников самой разной природы. Причем, эти данные обычно не имеют строгой внутренней структуры, в следствие чего их интеграция для использования в рамках некоторой единой ИС является нетривиальной задачей.

Большие объемы, высокая скорость накопления и отсутствие строгой внутренней структуры у «больших данных» также означают, что классические реляционные базы данных плохо подходят для их хранения. Тот факт, что доступ к «большим данным» во многих информационных системах осуществляется пакетно, или же в режиме реального времени, также делает реляционные БД не лучшим решением для хранения «больших данных». С другой стороны, распределенные ФС наподобие HDFS [13] или SageFS [14] и распределенные NoSQL [15, 16] БД хорошо подходят для решения задач хранения и передачи клиентам информационной системы больших объемов быстро накапливающихся или часто модифицируемых данных, не имеющих строгой внутренней структуризации – «больших данных».

В рамках данной работы будут преимущественно использоваться сравнительный и архитектурные подходы к определению «больших данных». Связано это с тем, что пространственные данные – особенно в растровых форматах – не полностью подходят под определения атрибутивного подхода, по крайней мере с точки зрения их конечного потребителя.

- **Разнообразие**: распространение пространственных данных осуществляется в ряде стандартизированных форматов с внутренней структурой данных. В силу этого, пространственные данные являются структурированными, а интеграция данных из различных источников значительно упрощена.
- **Достоверность**: бóльшая часть актуальных пространственных данных есть достоверная информация, поскольку они генерируются автоматизированными системами мониторинга окружающей среды. Исторические пространственные данные, однако, действительно могут быть существенно неточными.

3. Постановка задачи

Развитие технологий в современном мире позволяет человеку иметь доступ ко все бóльшим объемам данных из самых различных областей жизнедеятельности. И эколого-географический анализ не является исключением. Ежедневно и ежечасно цифровые каталоги информации об экологии, погоде, климате и распространении биологических объектов уточняются и дополняются усилиями сотен и тысяч людей и автоматических систем, ведущих наблюдение с поверхности Земли, с самолетов и атмосферных зондов, а также из космоса.

И рост объемов собранной информации постоянно ускоряется: растет как число источников данных, так и детализация самих данных. Пространственные данные в современном мире являются «большими данными». Как следствие, лабораториям и организациям, занимающимся анализом и обработкой пространственных данных необходимо приспосабливаться к этим новым реалиям. Это, в частности, означает переход от персональных настольных ГИС, едва ли способных эффективно работать с «большими данными», к распределенным системам, которые оптимизированы под обработку и хранение колоссальных объемов данных.

На сегодняшний день потребность в подобных распределенных географических информационных системах еще не удовлетворена, и специалисты, занимающиеся эколого-географическим анализом и моделированием, по-прежнему нуждаются в надежном и доступном инструменте для хранения и обработки больших объемов пространственных данных.

Удовлетворение потребности в подобной системе является актуальной задачей, и одним из решений может быть использование географических информационных систем, адаптированных под задачи эколого-географического моделирования и предназначенные для работы в добровольных и гибридных вычислительных сетях.

В данной работе будет рассмотрен прототип распределенной вычислительной системы, реализующий концепцию географической информационной системы для добровольных и гибридных вычислительных сетей. К данному прототипу могут быть выдвинуты следующие требования:

- Подобная распределенная ГИС должна уметь эффективно и надежно хранить пространственные «большие данные», принимая во внимание и применяя во благо исследовательского процесса особенности различных форматов хранения пространственных данных. В создаваемой ГИС должна быть гарантирована долговечность хранимых данных, а выполняемые операции должны обладать свойством атомарности. Данные в системе должны быть согласованными или, если это невозможно, согласованными в конечном счете.

- Создаваемая система является распределенной и должна стабильно работать в добровольных и гибридных вычислительных сетях. ГИС должна оперативно реагировать на изменения в количестве и степени загруженности входящих в состав сети узлов, динамически распределяя поступающие от пользователей задания между доступными и наиболее свободными узлами в сети.

- Создаваемая распределенная географическая информационная система должна уметь выполнять как минимум базовые операции обработки пространственных данных. В случае, когда пространственные данные представлены растровыми форматами, такими обязательными операциями являются операции растровой реклассификации и растровой алгебры. Операции должны быть реализованы в создаваемой системе таки образом, чтобы системой достигалась максимальная параллелизация в обработке пространственных «больших данных».

- Визуализация данных в создаваемой системе также должна осуществляться с максимальной параллелизацией. Кроме того, крайне желательной является возможность просмотра результатов визуализации, выполненной в создаваемой ГИС, при помощи других настольных или web ГИС.

При разработке описываемой географической информационной системы также является желательным использование свободно-распространяемого программного обеспечения.

4. Обзор области

Работа с «большими данными» является чрезвычайно-популярным направлением исследований, однако, лишь немногие из этих исследований посвящены вопросам работы именно с пространственными «большими данными».

Разумеется, все решения, разработанные для «больших данных» произвольной природы, будут успешно работать и для пространственных «больших данных». Но, в силу того, что они не учитывают особенности структуры и алгоритмов обработки пространственных данных, во многих случаях они окажутся менее эффективными, чем специализированные методы хранения и обработки пространственных «больших данных». Тем не менее, стоит выделить ряд работ, посвященных системам хранения и обработки «больших данных» произвольной природы, в которых были описаны интересные и новаторские идеи, которые могут быть успешно применены в системах, ориентированных на работу с пространственными данными:

1. «**Flat Datacenter Storage**» [17] представляет собой систему хранения данных в распределенной системе с высокой утилизацией и малым простоем дисков, и со статистической мультиплексией потоков данных.

   Система FDS предоставляет доступ к жестким дискам единообразно для всех процессоров вычислительного кластера, что дает возможность использовать динамическое распределение заданий и позволяет использовать максимально естественные модели вычислений без потери производительности. Кроме того, данная система отличается крайне высокой скоростью восстановления в случае сбоя дисков [18, 19]. Наконец, при использовании данной технологии существует возможность независимого апгрейда вычислительных ресурсов кластера.

   Использование технологии FDS вне зависимости от природы хранимых данных позволяет обеспечить их сохранность и быстрый доступ к ним.

   Технология Flat Datacenter Storage была разработана Эдмундом Найтингейлом (Edmund B. Nightingale) и его коллегами из университета Техаса в Аустине, и была впервые представлена на 10 симпозиуме USENIX в 2012 г.

2. Статья «**SageFS: the location aware wide area distributed filesystem**» [14] описывает методологию построения гибкой и простой в использовании сетевой файловой системы с запоминанием физического адреса данных (Location-Aware File System) и поддержкой репликации данных.

   SageFS по сути является надстройкой над локальными файловыми системами и предназначена для использования в гетерогенных вычислительных кластерах собранных из обычных персональных компьютеров. SageFS реализована в виде клиентской библиотеку на Python, что делает ее установку и обслуживание максимально простыми задачами. Помимо этого, благодаря малому размеру библиотеки и простоте используемых алгоритмов, SageFS имеет минимальное влияние на доступные вычислительные ресурсы узлов кластера.

   При этом, грамотное использование возможностей SageFS или эквивалентных сетевых файловых систем с запоминанием физического адреса данных делает возможным заметное повышение производительности кластера [20], особенно в случае сильного параллелизма алгоритмов обработки данных.

3. **MapReduce** [21] является моделью распределенных вычислений, используемой для обработки «больших данных» на вычислительных кластерах. Недостатком классической реализации данной модели является низкая производительность при работе в добровольных вычислительных сетях. Причина этого кроется в плохой приспособленности алгоритмов репликации данных к условиям частой недоступности отдельных узлов кластера.

   Попытка решения данной проблемы модели MapReduce была представлена в статье «**MOON: MapReduce On Opportunistic eNvironments**» [22].

   - **Репликация данных**: В классической реализации, HDFS достигает 99.99% доступности хранимых данных путем репликации. Однако, в добровольных вычислительных сетях узлы недоступны в среднем около 40% времени [22]. При таких условиях, HDFS требует создания как минимум 11 реплик для поддержания заданного уровня доступности данных. Поддержание такого большого числа реплик данных является непозволительной роскошью для абсолютного большинства добровольных вычислительных сетей.

     Система MOON предполагает наличие в вычислительной сети небольшого числа специализированных узлов с уровнем доступности 99.9% или более, основной задачей которых является хранение данных. При таком раскладе, 99.99% доступность хранимых данных может быть достигнута при всего лишь четырех репликациях, из которых только одна обязана находиться на специализированных узлах.

Дополнительно, специализированные узлы могут быть использованы для решения длительных или критически-важных задач, и эти задачи будут гарантировано выполнены.

- **Репликация промежуточных результатов**: Классический Hadoop не реплицирует промежуточные результаты вычислений. В силу этого, сбой узла в ходе вычислений приводит к потере промежуточных результатов и перезапуску процессов, необходимых для получения этих данных.

  В системе MOON используется многомерный сервис репликации который создает копии в том числе и промежуточных результатов. Несмотря на то, что промежуточные файлы обычно не имеют реплик на выделенных узлах вычислительного кластера, это все равно заметно уменьшает вероятность того, что промежуточные результаты будут безвозвратно потеряны при сбое узла и потребуется повторное проведение вычисление.

- **Планирование задач**: В классической реализации системы Hadoop при планировании задач предполагается, что абсолютное большинство из них успешно завершится без каких-либо эксцессов. К сожалению, это не всегда возможно в добровольных вычислительных сетях, и стандартная стратегия репликация заданий Hadoop не справляется с высокой волатильностью.

  В системе MOON состояние выполняемых задач оценивается более гибко, что позволяет различать реально зависшие задачи и задачи, выполнение которых замедлилось или было временно приостановлено из-за возросшей активности локального пользователя добровольного узла.

  Помимо этого, в системе MOON, при достижению некоторого заданного процента выполнения задания, начинается прогрессивная репликация еще не завершенных заданий с целью максимально-быстрого гарантированного прохождения «финишной прямой» вычислений.

  Наконец, как уже было упомянуто выше, если задача является критически важной или время ее выполнения превышает среднее время доступности добровольного узла, то такая задача может быть запущена на выделенных узлах, что будет гарантировать ее успешное выполнение.

В ходе проведенных экспериментов, модель распределенных вычислений MOON показала сравнимые с классическим Hadoop результаты в случае, когда узлы вычислительной сети доступны 90% времени, и выполняла задачи почти вдвое быстрее, когда узлы вычислительной сети доступны только 50% времени [22]. При этом, в ходе экспериментов система MOON порождала заметно меньшее число дублированных задач, чем классический Hadoop.

4. В статье «**Osprey: Implementing MapReduce-Style Fault Tolerance in a Shared-Nothing Distributed Database**» [23] повествуется о путях повышения устойчивости распределенных хранилищах данных к возникающим ошибкам. Авторы статьи показывают, что перезапуск запросов в OLAP системах в случае возникновении сбоев является дорогостоящей операцией. Особенно когда такая система должна выдавать отчеты в строго-оговоренные интервалы времени.

   Для разрешения данной проблемы OLAP-систем, авторами статьи предлагается использовать обработку запросов, реализующую следующие ключевые идеи:

   - Связующее программное обеспечение системы разделяет низкоуровневые задачи – такие как непосредственное выполнение SQL запросов – и задачи высокого уровня, связанные с обеспечением отказоустойчивости хранилища данных в целом.
   - Устойчивость к сбоям узлов системы достигается путем репликации данных с использованием цепной декластеризации, и разбиением поступающих в систему запросов на независимые подзапросы, которые могут исполняться независимо друг от друга на одной или более репликах данных. Задача отслеживания процесса выполнения запроса возлагается на координатора, который играет роль ведущего устройства.
   - Балансировка нагрузки на OLAP-систему осуществляется динамическим распределением заданий.

   Созданное в соответствие с приведенными идеями хранилище данных «Osprey» показало себя как хорошо масштабируемое решение с практически идеальной балансировкой нагрузки при наличии доступных реплик данных. Накладные расходы на выполнение запросов также минимальны. Наконец, динамическое распределение заданий и разбиение запросов на подзапросы меньшего размера дают «Osprey» высокую устойчивость к возникающим в ходе работы сбоями.

5. Традиционные решения по хранению и обработке «больших данных» вроде MapReduce [21] и Hadoop [13] используют распределенные файловые системы для хранения данных. Это является естественным следствием того факта, что на момент их появления жесткие диски являлись единственными экономически-жизнеспособными устройствами с произвольным доступом, которые могли быть использованы для хранения «больших данных». Однако, прогресс не стоит на месте, и в настоящее время распределенная оперативная память может быть альтернативой распределенным файловым системам.

   В 2010 г. алгоритмы MapReduce были полностью реализованы в распределенной оперативной памяти в фреймворке MRlite [24]. Такая система обладает заметно более высоким быстродействием, однако имеет свои недостатки:

- Цена аппаратного обеспечения: оперативная память на два порядка дороже жестких дисков того же объема.
- Волатильность среды: выход сервера из строя приводит к полной потере данных, хранившихся в оперативной памяти данного сервера. Репликация данных может частично решить данную проблему, но это – дорогостоящий процесс, который приводит к резкому повышению латентности даже при репликации в рамках одного дата-центра.
- В облачных вычислениях, файловая система не только хранит данные, но и предоставляет глобальное пространство имен для коммуникации процессов, обеспечивает атомарность данных и обеспечивает контроль параллелизма в системе. Оперативная память в настоящее время не может предоставлять данные семантики.

Очевидным решением является создание гибридной системы, которая позволит сочетать быстродействие DRAM-фреймворков и семантики, предоставляемые распределенными файловыми системами в классических реализациях модели вычислений MapReduce. Построение такой гибридной системы описано в статье «**MapReduce-style Computation in Distributed Virtual Memory**» [25].

В статье описывается гибридная распределенная система Virtual On-Line Unified Memory Environment, которая объединяет оперативную и постоянную память узлов вычислительного кластера. По-умолчанию, данные в VOLUME хранятся в распределенной оперативной памяти, но существует возможность их записи на жесткие диски. Масштабирование системы и оптимизация потоков данных в VOLUME прозрачны для программиста. Также, VOLUME задает семантики для поддержания атомарности, изоляции и долговечности данных операций, производимых в распределенной оперативной памяти.

По результатам тестов, сортировка в VOLUME выполняется на 40% быстрее, чем при использовании классической модели MapReduce. Задачи подсчета слов в среднем выполняются на 36% быстрее и имеют лучшую масштабируемость по числу вычислительных узлов. K-means кластеризация выполняются в среднем в 7.55 раз быстрее, а построение списка смежности вершин графа выполняется до 11.7 раз быстрее, чем на Hadoop, обладая при этом сверх-линейным ускорением при увеличении числа вычислительных узлов [25].

Таким образом, использование VOLUME или эквивалентных гибридных систем позволяет добиться значительного прироста в скорости обработки «больших данных», присущего DRAM-фреймворками, не теряя при этом такое важное свойство данных, как долговечность. Такие системы могут быть использованы для интерактивных вычислений или обработки данных в реальном времени.

6. Для достижения максимальной эффективности обработки «больших данных» на вычислительном кластере, крайне важным является распределение заданий между узлами кластера таким образом, чтобы как можно большее число узлов получило задания на обработку данных, хранящихся локально.

   MapReduce [21] является системой, где задания распределяются между узлами кластера подобным образом. Однако, данная модель вычислений не является оптимальной для всех задач, которые требуют обработки «больших данных». У программистов должна быть возможность построения программы в терминах той модели вычислений, которая наилучшим образом подходит для решения их задачи. И, вне зависимости от того, какая модель вычислений была выбрана, у программы должна существовать возможность интеллектуальным образом распределять задания между узлами кластера.

   Система управления кластером Tashi, о которой рассказывается в статье «**Tashi: Location-aware Cluster Management**» [26], занимается администрированием контейнеров виртуальных машин пользователей кластера и, по возможности, распределяет поступающие от пользователей кластера задания по узлам так, чтобы как можно большее число узлов получило задания на обработку данных, хранящихся локально. Система Tashi является нейтральной по отношению к гипервизорам и распределенным файловым системам, в силу чего она может быть использована с большим числом различных конфигураций программного обеспечения.

   По результатам тестирования, приведенным авторами статьи, использование системы управления кластерами Tashi позволяет добиться ускорения обработки данных до 3.5 раза при хранении данных на жестких дисках и до 9 раз в случае использования твердотельных накопителей [26] по сравнению с распределением заданий случайным образом вне зависимости от выбранной модели вычислений.

В статьях, посвященных обработке пространственных «больших данных», чаще всего рассматривается вопрос обработки сенсорной информации с геопривязкой в режиме реального времени. Это решение таких задач, как мониторинг ситуации на дорогах, слежение за развитием разрушительных погодных явлений, слежение за облаком мусора на орбите Земли, и многое другое.

Системы, описанные в этих статьях, в качестве источника данных используют поток информации, предоставляемый множеством сенсоров различной природы. В силу этого, вопросам обработки статичных пространственных данных, таких как растровые и векторные слои с агрегированными значениями некоторых параметров по рассматриваемой области, уделяется мало внимания. Тем не менее, в этих статьях имеет место множество интересных наработок:

1. Директива INSPIRE призывает руководства стран Европейского Сообщества совместно размещать в свободном доступе тематические наборы уточненных пространственных данных. В проекте INSPIRE используется распределенная сервис-ориентированная архитектура, что делает его похожим на более ранний проект, Linked Open Data initiative (LOD). Эти проекты также ориентированы на хранение пространственных данных различных тематик, что делает идею совместного использования данных репозиториев крайне привлекательной.

   Однако, в проектах INSPIRE [27] и LOD используются различные подходы к описанию хранимых данных. В проекте LOD используются OWL-модели, а INSPIRE опирается на ISO и OGC стандартизированный подход и использует язык GML для описания объектов и их свойств. Трансляция запросов к одной системе в формат, который будет понят другой есть нетривиальная задача.

   В статье «**Semantic access to INSPIRE: how to publish and query advanced GML data**» [28] представлен транслятор запросов, позволяющий выполнять преобразования между форматами описаниями данных OWL и GML.

   Использование данного или эквивалентного транслятора позволяет создавать федеративные базы пространственных данных поверх систем, использующих GML и OWL форматы описания данных, и позволяет пользователям системы использовать без дополнительных затрат бóльшие объемы пространственных данных в проводимых исследованиях, повышая таким образом качество работ.

2. Кластеризация и анализ векторных данных является важным классом задач по обработке векторных «больших данных». Решение этих задач необходимо для успешной работы правоохранительных органов и аварийно-спасательных служб, мониторинга распространения инфекционных заболеваний, предсказания развития разрушительных погодных явлений и многого другого.

   И, во многих случаях, у специалистов существует потребность в доступе не только к самим данным, но и к различным инструментам по их анализу, в полевых условиях.

   Web-ГИС TerraFly GeoCloud, о которой рассказывается в статье «**TerraFly GeoCloud: An Online Spatial Data Analysis and Visualization System**» [29] является распределенной географической информационной системой, которая предоставляет своим пользователям возможность быстро и просто отображать, модифицировать и анализировать любые объемы пространственных данных в векторных форматах.

   TerraFly GeoCloud является online-инструментом, сочетающим в себе простоту пользовательского интерфейса и широкие аналитические возможности. Одним из ее единственных недостатков является ориентированность на работу только

с векторными данными: данная ГИС не обладает возможностями растрового анализа. Несмотря на этот недостаток, TerraFly GeoCloud является примером того, как может выглядеть и работать научная web-ГИС.

3. Как уже было сказано ранее, возможность взаимодействия между различными системами по обработке пространственных данных делает возможным решение значительно более широкого спектра задач. Однако, в силу того, что различные географические информационные системы используют в своей работе разные способы представления пространственных данных, задача создания протоколов взаимодействия между различными ГИС является нетривиальной.

   В статье «**Geospatial Semantics: Why, of What, and How**» [30] приводится список возможных проблем, с которыми можно столкнуться в ходе настройки семантической совместимости между различными ГИС, а также предлагается фреймворк для решения данных проблем.

   Использование наработок, описанных в данной статье позволяет значительно расширить возможности географических информационных систем благодаря реализации эффективного взаимодействия между ними как при хранении и визуализации данных, так и при их анализе и обработке.

Пространственные данные, представленные растровыми форматами, по своей сути являются изображениями, каждый пиксель которых задается не набором значений основных цветов – например, красного, зеленого и синего – а одним единственным числовым значением, соответствующим среднему значению некоторого параметра на территории, покрываемой данным «пикселем».

В силу этого, многие из техник по обработке больших растровых изображений могут быть применены и к обработке растровых пространственных данных.

1. Одной из наиболее часто встречающихся проблем, возникающих при обработке «больших данных», является потребность в значительных объемах оперативной памяти, необходимой, чтобы операции над данными выполнялись за разумное время. Это является верным и для больших графических данных, таких как, например, серии медицинских снимков со сверх-большим разрешением.

   Одно из решений данной проблемы было предложено командой исследователей из Pivotal's Data Science Labs [31]. Их предложение заключается в том, что после преобразования в табличный формат, графические изображения могут быть обработаны средствами распределенной базы данных. Это делает возможным выполнение таких «тяжелых» операций, как поиск и идентификация объектов на изображении, даже на небольших добровольных вычислительных кластерах с ограниченными объемами доступной оперативной памяти.

2. Графические процессоры являются крайне эффективными устройствами для обработки графических данных. И в силу того, что пространственные данные в растровых форматах по своей сути графическими объектами, использование графических процессоров для их обработки является разумной идеей. Однако, «большие данные» потому и называются большими, их обработка на одном отдельно-взятом компьютере является крайне трудоемкой задачей.

   Поэтому компьютерная система, используемая для обработки сверх-больших изображений, должна быть вычислительным кластером или облаком даже если для обработки изображений используются графические процессоры.

   В статье «**Large-Scale Image Processing Research Cloud**» [32] описывается распределенная вычислительная система на основе hadoop-кластера, которая, благодаря использованию библиотеки OpenCL, может проводить вычисления, используя не только центральные процессора, но также и другие процессора и аппаратные ускорители.

   В соответствие с полученными в ходе экспериментов результатами, подобный вычислительный кластер выполняет различного рода алгоритмы по анализу растровых изображений за значительно меньшее время, чем системы, в которых используется последовательной обработкой изображений.

   В экспериментах, описанных в статье [32] использовался кластер из одного управляющего и семью вычислительных узлов, каждый из которых обладает 16 ядрами процессора и 128 Гб памяти.

	малые изображения 5425 файлов (594 Мб)	средние изображения 2539 файлов (3621 Мб)	большие изображения 400 файлов (4436 Мб)
последовательная обработка	1386.02 секунды	4511.35 секунд	5716.31 секунды
параллельная обработка на hadoop	228 секунд	140 секунд	97 секунд

Таблица 2: Сравнение времени выполнения алгоритма распознавания лиц

	соотнесение с шаблоном	DFT	распознавание лиц
последовательная обработка	860 секунд	2207 секунд	5716 секунд
параллельная обработка на hadoop	156 секунд	105 секунд	97 секунд
достигнутое ускорение	5.5 раз	21 раз	59 раз

Таблица 3: Сравнение времени выполнения различных алгоритмов анализа изображения

В соответствие с приведенными в статье [32] расчетами, в идеальных условиях, достигнутое ускорение может быть даже больше:

Пусть N есть число изображений, T_i – время на выполнение I/O операций для одного изображения, T_d – время на кодирование и раскодирование одного изображения, а T_a – время выполнения алгоритма для одного изображения. Тогда, общее время на последовательную обработку набора изображений может быть представлено формулой:

$$T_s = (T_i + T_d + T_a) \times N$$

Пусть, T_p есть время, затраченное на создание задачи hadoop, T_m – среднее время выполнения операции map, что примерно соответствует значению суммы $T_i + T_d + T_a$, T_r – среднее время отчета mapper'а, а T_c – время, потраченное на очистку системы после выполнения задач. Тогда, если положить, что C есть число используемых процессорных ядер, общее время обработки изображений на hadoop-кластере может быть описано формулой:

$$T_h = T_p + (T_m + T_r) \times (N \div C) + T_c$$

Соответственно, ускорение вычислений будет описывается формулой:

$$S = \frac{T_s}{T_h} = \frac{(T_i + T_d + T_a) \times N}{T_p + (T_m + T_r) \times (N \div C) + T_c}$$

В ходе экспериментов, описанных в статье [32], T_p составляло 8 секунд, а T_c было равно 3 секундам. Таким образом, при идеальных условиях, для маленьких изображений может быть достигнуто ускорение:

$$S = \frac{1386.02}{8 + (0.2555 + 2.5) \times \lceil \frac{5425}{112} \rceil + 3} = \frac{1386.02}{147.2} = 9.4 \text{ раза}$$

А для изображений большого размера:

$$S = \frac{5716.31}{8 + (14.3 + 2.5) \times \lceil \frac{400}{112} \rceil + 3} = \frac{5716.31}{87.65} = 65 \text{ раз}$$

Таким образом, использование hadoop-кластеров для обработки изображений, позволяет значительно сократить время выполнения операций над данными изображениями. А поскольку пространственные данные в растровых форматах являются по своей сути графическими изображениями, данная техника может быть применена и их обработке.

5. Распределенная ГИС для работы с «большими данными»

Как уже было сказано во введении, развитие технологий привело к тому, что в современном мире объемы пространственных данных, особенно представленных растровыми форматами, растут с огромной скоростью. И это касается не только количества файлов; размер самих файлов, содержащих в себе пространственные данные, также постоянно увеличивается.

Как следствие этого, пространственные данные в растровых форматах все более часто становятся «большими данными», а их обработка на одиночных компьютерах становится трудно-выполнимой за «разумное» время задачей, особенно в свете того, что многие из лабораторий, занимающихся анализом пространственных данных, не имеют доступа к современным высокопроизводительным рабочим станциям.

При этом, использование сторонних облачных ресурсов, таких как, например, TerraFly GeoCloud [29] или ее аналогов, для выполнения ресурсоемких операций по обработке и анализу пространственных данных, может оказаться невозможным в силу внутренней политики лаборатории или по иным причинам. С другой стороны, классические кластерные решения, вроде Hadoop MapReduce [21], плохо подходят для использования в добровольных вычислительных сетях. А, к сожалению, малые лаборатории, занимающиеся анализом пространственных данных, скорее всего будут обладать именно такими вычислительными сетями.

Таким образом, распределенные Географические Информационные Системы с аналитическими возможностями, способные эффективно выполнять обработку и анализ «больших данных» в гетерогенных добровольных вычислительных сетях остаются по-прежнему востребованными.

В данной работе предлагается проект географической информационной системы для работы в добровольных вычислительных сетях, обладающей возможностями обработки и анализа пространственных данных. К подобной системе выдвигаются следующие требования:

1. ГИС должна эффективно работать в добровольных вычислительных сетях и ее работоспособность должна сохраняться даже в ситуациях, когда часть узлов вычислительной сети являются недоступными.

2. ГИС должна работать в гетерогенных вычислительных сетях.

3. Для достижения полной и гарантированной доступности хранимых данных, ГИС должна использовать минимум аппаратных ресурсов сверх выделяемых ей добровольной вычислительной сетью.

4. Распределенная Географическая Информационная Система должна обладать аналитическими возможностями, сопоставимыми с возможностями настольных ГИС. Анализ пространственных «больших данных» в распределенной системе должен выполняться быстрее, чем в настольных ГИС. Следующие операции растрового анализа должны быть реализованы в распределенной ГИС:

   - Растровая реклассификация – при выполнении данной операции исходные значения точек растра замещаются новыми в соответствие с задаваемой пользователем ГИС таблицей.

     Растровая реклассификация применяется для выделения областей растра, значения в которых удовлетворяют некоторому критерию. Данная операция является одной из основных операций эколого-географического анализа.

     Растровая реклассификация принимает в качестве своих аргументов один растровый слой и таблицу преобразования значений, и возвращает новый растровый слой при успешном своем выполнении.

   - Растровая алгебра – при выполнении данной операции, значения точек на растре преобразуются в соответствие с задаваемой пользователем системы формулой. В данной формуле могут быть использованы арифметические операции, элементарные функции, константы и значения соответствующих точек других растровых слоев.

     Растровая алгебра применяется для построения растров, представляющих характеристики территории, которые не могут быть измерены напрямую (напр., гидротермический коэффициент увлажнения), а также для поиска территорий, одновременно удовлетворяющих двум или более критериям. Растровая алгебра также является одной из основных операций эколого-географического анализа.

     Растровая алгебра принимает формулу в качестве аргумента и возвращает новый растровый слой при своем успешном выполнении

   - Перепроецирование – вспомогательная операция, позволяющая перевести слой из одной географической проекции в другую.

Дополнительно, большим плюсом для распределенной ГИС будет возможность ее функционирования в качестве WMS-сервиса, поскольку это позволит значительно упростить демонстрацию получаемых результатов путем исключения необходимости в передаче больших файлов на удаленные компьютеры.

А именно, использование WMS-сервис позволяет показывать пространственные данные в виде интерактивной карты, в каждый момент времени отображающей только набор изображений (т.н. тайлов). Такой набор изображений «весит» на один или более порядков меньше, чем исходный файл слоя. Информация о значения слоя в интересующих пользователя точках также предоставляется через WMS-сервис.

5.1. Хранение данных

Географические Информационные Системы в большинстве случаев занимаются обработкой и анализом пространственных данных, собираемых за продолжительные периоды времени. Как следствие этого, имеет место задача эффективного хранения данных, которые обрабатываются в настоящее время или будут обрабатываться в будущем. К сожалению, растры с высоким пространственным разрешением, равно как и векторные слои с высокой детализацией или большим количеством элементов, представляются очень «тяжелыми» файлами. Помимо этого, отображение подобных слоев с малым приближением требует больших затрат вычислительных ресурсов: в случае растровых слоев – это затраты на масштабирование растра; для векторных слоев – это отрисовка большого числа точек или линий.

Проблемы с чрезмерными размерами файлов пространственных данных и/или сложностями их представления пользователям системы могут быть частично или полностью разрешены благодаря использованию определенных техник организации пространственных данных в постоянной памяти. Для пространственных данных в растровых форматах такими техниками являются «пирамида» [33] и «мозаика» [34].

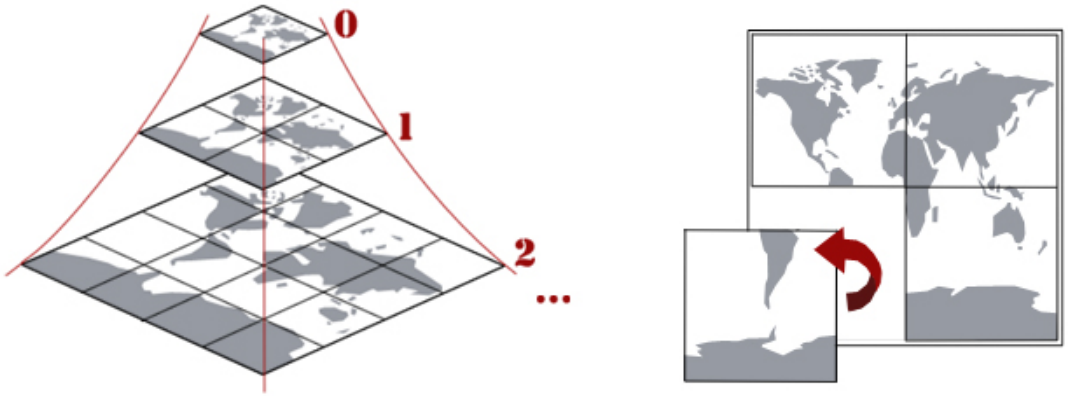

Рис. 1: Техники организации растровых данных на диске: пирамида (слева) и мозаика (справа)

5.1.1. Пирамида

Пирамида является классической техникой для уменьшения вычислительных нагрузок на обрабатывающий узел при визуализации пространственных данных в растровых форматах и заключается в хранении дополнительных копий растра с меньшим пространственным разрешением в качестве избыточных данных. Обычно, ГИС используют уровень приближения равный 2^n. Если принять, что меньшая из сторон растра имеет N точек, то пирамида будет иметь не более $log_2 N$ уровней. На самом деле, уровней в пирамиде будет гораздо меньше, так как даже самый верхний из уровней должен являться полноценным растром, а не вырождаться в точку или линию. Тем не менее, даже в худшем случае, пирамида увеличивает количество хранимых данных в:

$$E = 1 + \sum_{k=1}^{log_2 N} \frac{1}{2^{2 \times k}} \approx 1.3(3) \text{ раза}$$

Пирамида обычно не имеет смысла для растровых слоев с разрешением меньше 25 МПикс. Для слоев же с бóльшим разрешением разрешением, пирамида позволяет заметно уменьшить время формирования тайлов с большим удалением. Некоторые файловые форматы для хранения растровых пространственных данных (например, формат BigTiff [35, 36]) имеют встроенную поддержку пирамиды.

Аналогичные техники могут быть применены и для пространственных данных в векторных форматах: в дополнение к основному слою могут также храниться его «снимки» с различными уровнями кластеризации и сглаживания элементов. Этот подход максимально эффективен для точечных векторных слоев, поскольку данные с геопривязкой к единственной точке максимально подходят для кластеризации, но он может быть использован и для «мультилиний» и «мультиполигонов».

5.1.2. Мозаика

Мозаикой называется техника при которой растровый слой разбивается на набор слоев меньшего размера, в сумме покрывающих ту же территорию, что и исходный слой. Мозаика часто используется для того, чтобы минимизировать использование оперативной памяти при работе с большими слоями, но также может использоваться и для того, чтобы распределить растр с высоким пространственным разрешением по нескольким узлам вычислительной сети.

Такое распределение элементов растровой мозаики по узлам вычислительной сети является допустимым, так как большая часть алгоритмов растрового анализа, включая растровую реклассификацию и алгебру, в ходе своей работы не используют

значения соседних точек растра. Иначе говоря, большинство алгоритмов растрового анализа при рассмотрении точки с координатами (X, Y) не учитывают значения в точках с координатами $(X \pm \Delta, Y \pm \Delta)$, и, как следствие, N узлов вычислительной сети могут одновременно выполнять анализ N элементов мозаики.

Эквивалентом растровой мозаики для векторных слоев может служить обычное разбиение массива элемента слоя на несколько раздельно-хранящихся наборов. В отличие от растров и точечных векторов, однако, разбиение «мультилинии» или «мультиполигона» на мозаику из примерно одинаковых по объемам потребляемой памяти элементов может оказаться невозможным без раздробления одного или более примитивов на несколько частей. Кроме того, если не дробить линии или полигоны на части, то во многих случаях будет невозможно разделить векторный слой на мозаику из непересекающихся прямоугольных областей. Это, однако, не является большой проблемой для векторных слоев, в отличие от растров, которые всегда должны покрывать прямоугольную область на карте.

Мозаика также может использоваться параллельно с пирамидой. В этом случае, техника мозаики будет применяться к каждому из уровней пирамиды, кроме самого верхнего, дробить который далее не имеет смысла. При распределении элементов мозаики по нескольким узлам вычислительной сети, имеет смысл создавать копии самого верхнего уровня пирамиды на каждом из используемых узлов сети – этот элемент мозаики является наиболее часто запрашиваемым и такое «кэширование» позволит быстрее получать доступ к нему.

5.1.3. Распределение данных по узлам сети

Как было сказано в предыдущем разделе, обработка пространственных данных, в особенности растровых, прекрасно параллелизуется в силу того, что большинство алгоритмов обработки пространственных данных в ходе своей работе используют значения текущего элемента и не используют значения его «соседей». Поэтому, при работе с пространственными данными разумным является распределение хранимых данных по узлам сети, при котором соседствующие элементы мозаики находятся на различных узлах этой вычислительной сети.

Для растровых слоев это означает, что любые 4 элемента мозаики, которые вместе образуют прямоугольник 2×2 должны быть распределены по различным узлам вычислительной сети. Если это условие выполнено, то вне зависимости от того, с какой областью работает пользователь, ГИС сможет максимально распараллелить поступающие запросы на обработку, анализ или визуализацию данных. Например, для вычислительной сети из четырех выделенных узлов, следующее распределение элементов мозаики трехуровневой пирамиды будет одним из оптимальных:

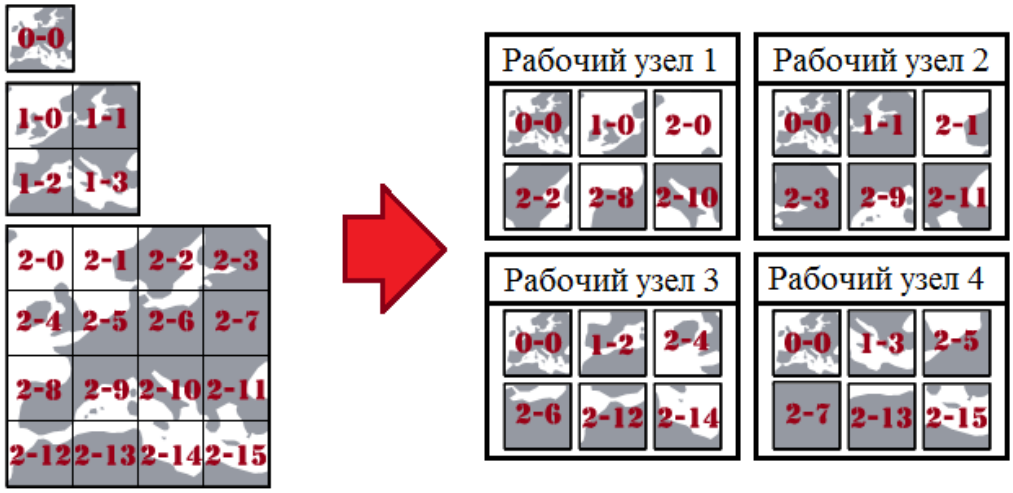

Рис. 2: Распределение элементов мозаики по узлам вычислительной сети

При распределении элементов растровой мозаики по узлам вычислительной сети, подобному тому, что приведен на рис. 2, для любого из уровней пирамиды, кроме самого нижнего, любой запрос пользователя на обработку или визуализацию данных расположенных на более чем одном элементе мозаики, может быть распараллелен. Обработка слоя целиком также будет выполняться с максимальной параллелизацией для данного числа выделенных узлов в вычислительной сети.

Если же вычислительная сеть является добровольной и входящие в нее узлы оказываются недоступными часто и на продолжительное время, то, естественно, требуется множество копий для гарантии 99.9% доступности данных. В случае когда узлы сети доступны только 60% времени, указанных уровень доступности данных требует поддержания аж 11 копий [22]. И если каждая копия растровой мозаики рабиваемтся на 4 части, то для хранения 11 копий потребуется аж 44 узла. Что, с высокой долей вероятности значительно превосходит количество компьютеров в небольшой эколого-географической лаборатории.

Наиболее оптимальным выходом из данной ситуации видится добавление в состав добровольной вычислительной сети группы выделенных узлов, на которые будет возложена задача предоставления пользователям всегда доступных копии данных и некоторых вычислительных мощностей. Эти ресурсы будут использоваться только когда отсутствует возможность использования ресурсов добровольных узлов. При такой организации вычислительной сети для достижения гарантированной 99.9% доступности данных требуется содержать всего 4 копии данных: 3 на добровольных узлах и 1 на выделенных [22]. В реальности, возможно обойтись и меньшим числом копий данных. Поддержание гибридной вычислительной сети из 12 – 16 узлов уже не является чем-то непосильным даже для относительной небольшой лаборатории.

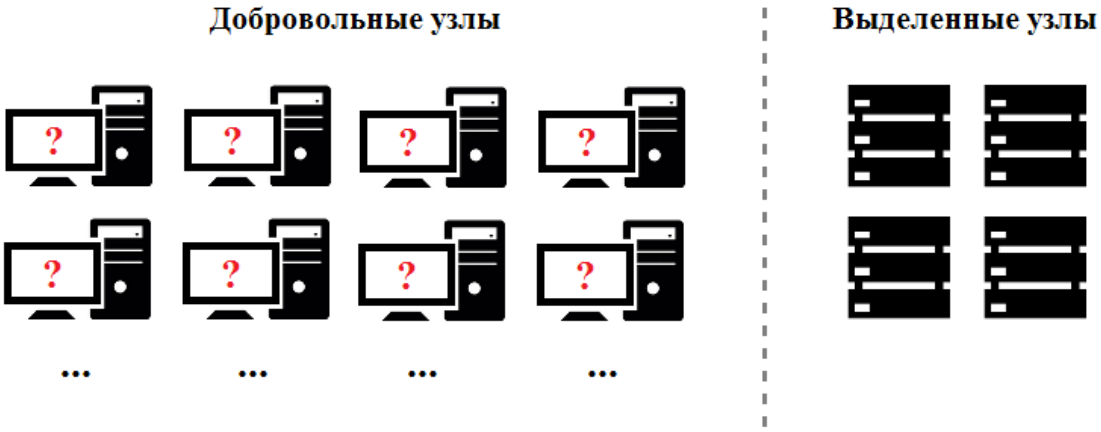

Рис. 3: Гибридная вычислительная сеть.

В такой гибридной сети хранимые данные всегда будут иметь копию, хранимую на выделенных узлах. Дополнительные копии этих данных могут – но, на самом деле, не обязаны – существовать на добровольных узлах. При чтении данных в такой сети, сначала ищутся копии на добровольных узлах, и только в случае если таких копий не обнаружено, система будет обращаться к выделенным узлам. При записи данных, наоборот, сначала создается копия на выделенных узлах, после чего уже идет создание дополнительных копий на добровольных узлах сети.

Данный подход к организации чтения / записи данных в вычислительной сети имеет своей целью обеспечить максимальную долговечность данных и, одновременно с этим, максимально использовать ресурсы добровольных узлов при чтении данных.

Если по каким-то обстоятельствам на добровольных узлах вычислительной сети присутствует более новая версия данных, чем хранимая на выделенных узлах, то система должна определить, возможно ли «собрать» полную мозаику слоя новой версии. Если ответ положительный, копия слоя на выделенных узлах подменяется версией с добровольных узлов, после чего обновляются остальные копии данных на добровольных узлах. В противном случае же система выполнит откат данных на добровольных узлах к версии, хранимой на выделенных узлах.

Данный подход к работе с различными версиями данных делает распределенную ГИС системой с согласованностью данных в конечном счете [37, 38]: хотя в системе возможны ситуации, при которых на различных группах вычислительных узлов одновременно существуют различные версии одного слоя, в обозримом будущем все копии данных, хранящихся в системе, будут иметь последнюю стабильную версию.

В ситуации, когда на добровольных узлах вычислительной сети существует две или более конфликтующие версии данных, решение о том, какая из них является актуальной, может приниматься на основе временных меток, или же оно может быть возложено на администратора сети.

5.2. Обработка данных

Как уже было сказано в начале этой главы, распределенная ГИС для добровольных вычислительных сетей должна поддерживать как минимум следующие операции по обработке пространственных данных в растровых форматах:

- Перепроецирование
- Растровая реклассификация
- Растровая алгебра

Иными словами, ГИС должна уметь изменять географическую проекцию слоев, в том числе разбитых на мозаику; выделять на растрах области, соответствующие задаваемым пользователем критериям; а также изменять значения точек на растрах в соответствие с задаваемыми пользователями формулами.

Данные операции по обработке пространственных данных в растровых форматах будут далее рассмотрены более подробно:

5.2.1. Перепроецирование

Картографической проекцией называется систематический способ переноса широт и долгот объектов на поверхности геоида, для простоты считаемого эллипсоидом вращения, на плоскость. Однако, поскольку эллипсоид не может быть развернут на плоскость, результат проецирования всегда будет иметь искажения. Эти искажения могут быть четырех видов:

1. **Искажения длин**: базовое искажение, означающее непостоянство масштаба плоского изображения. Это проявляется в изменении масштаба карты от точки к точке, и даже в одной и той же точке в зависимости от направления. Это означает, что на карте присутствует два вида масштаба:
   - Главный – это масштаб исходного эллипсоида, развертыванием которого в плоскость карта и получена.
   - Частный масштаб – их бесконечно много, и они меняются от точки к точке и в пределах любого сколь угодно малого объекта.

   Для наглядного изображения частных масштабов вводят эллипс искажения.
2. **Искажение площадей**: логически вытекает из искажения длин в проекции. За характеристику искажения площадей принимают отклонение площади эллипса искажений от исходной площади на эллипсоиде.

3. **Искажение углов**: логически вытекает из искажения длин. За характеристику искажений углов на карте принимают разность углов между направлениями на карте и соответствующими направлениями на поверхности эллипсоида.

4. **Искажение формы**: графическое изображение вытянутости эллипсоида.

Поскольку «идеальной» проекции не существует и что-то всегда будет искажено, существует множество проекций, каждая из которых оптимизирована для решения определенной задачи: например, для нужд навигации используются равноугольные проекции, которые сохраняют углы, но сильно искажают площади и формы [39].

Задачей **перепроецирования** является приведение пространственных данных из их исходной картографической проекции в проекцию, которая является оптимальной для решения некоторой задачи с точки зрения используемых алгоритмов или просто здравого смысла. Точно также, если необходимые для решения некоторой задачи данные представлены в различных проекциях, то первым шагом в решении данной задачи должно быть приведение всех данных к некоторой общей проекции.

Задача перепроецирования данных является хорошо изученной и для ее решения созданы специализированные библиотеки. Одной из наиболее полных библиотек является библиотека GDAL – Geospatial Data Abstraction Library [40]. Эта библиотека является свободно-распространяемым ПО и активно поддерживается сообществом. В библиотеку GDAL заложено порядка пять тысяч картографических проекций и она умеет работать с 142 растровыми и 84 векторными форматами данных. Все это вместе делает данную библиотеку оптимальным выбором в качестве модуля по перепроецированию данных для создаваемой ГИС.

5.2.2. Растровая реклассификация

Операция растровой реклассификации является одной из основных операций в растровом анализе и широко используется при проведении эколого-географического анализа и моделирования. Данная операция используется для определения областей, значения точек растра в которых лежит в пределах задаваемых пользователем ГИС границ. Например, растровая реклассификация может быть использована для того, чтобы найти на карте среднемесячных температур за январь месяц все территории, на которых значение температуры оказалось выше $-20\,^{\circ}\mathrm{C}$.

При выполнении эколого-географического анализа и моделирования, растровая реклассификация используется для выделения на растрах экологически-пригодных территорий по какому-либо из значащих факторов. Такими значащими факторами для биологических объектов могут быть температура, среднемесячное количество осадков, гидротермический коэффициент, засоление почвы и многие другие.

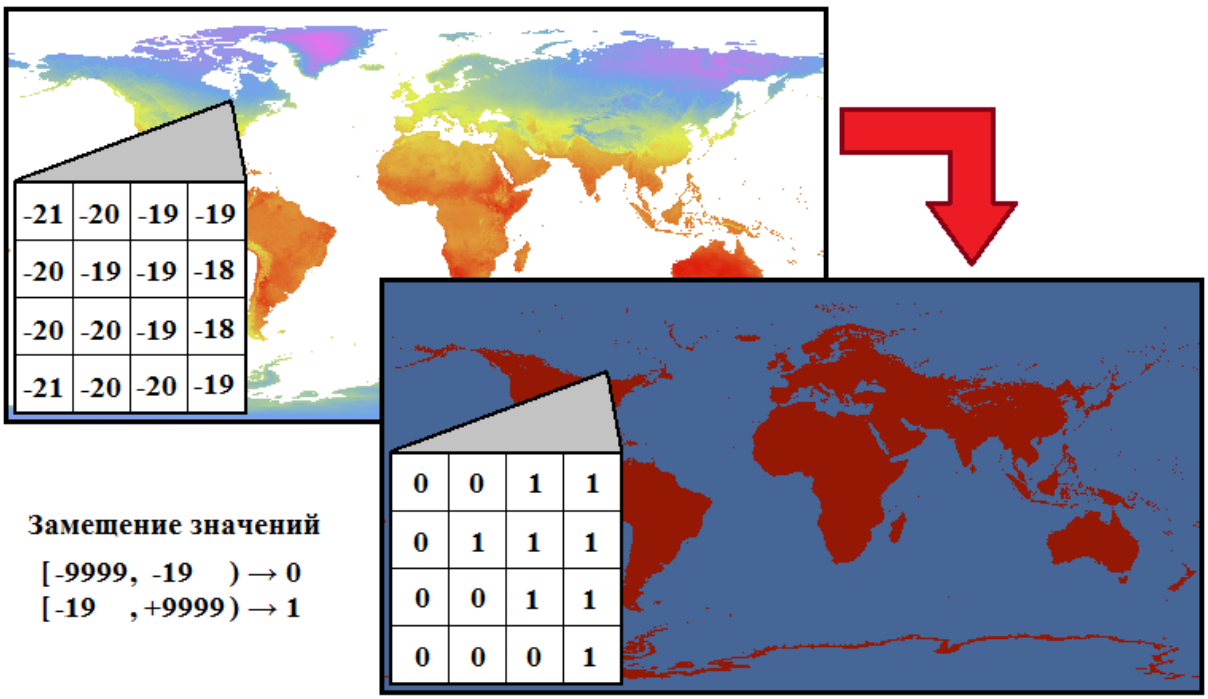

Рис. 4: Растровая реклассификация

Операция растровой реклассификации принимает в качестве своих аргументов один растровый слой и таблицу, в соответствие с которой будут замещаться значения на данном растре. Результатом выполнения растровой реклассификации является новый слой с теми же разрешением, картографической проекцией и параллельным осям ограничивающим прямоугольником (bounding box), что и исходный слой.

Пусть L – это множество растровых слоев, а V – множество таблиц замещения значений, тогда растровая реклассификация может быть представлена функцией:

$$f\colon L, V \to L$$

Таблицы замещения значений представляют собой непустые множества наборов из трех действительных чисел x_1, x_2 и y, где y есть значение, которое будет присвоено всем точкам исходного растра, значения которых лежат в интервале $[x_1, x_2)$. В случае если значение какой-либо точки исходного растра не попало ни в один интервалов из таблицы замещений, значение этой точки переносится на новый растр без изменений.

Поскольку растровая реклассификация рассматривает каждую точку растрового слоя независимо от остальных, все элементы мозаики растра могут обрабатываться параллельно. Иными словами, реклассификация «больших» растровых слоев может быть значительно ускорена путем разбиения их на отдельные небольшие фрагменты, каждый из которых обрабатывает отдельным узлом вычислительной сети.

5.2.3. Растровая алгебра

Операция растровой алгебры является одной из основных операций в растровом анализе и широко используется при проведении эколого-географического анализа и моделирования. Данная операция используется в основном для решения следующих двух задач:

1. Генерация растровых слоев, содержащих информацию о значениях какой-либо квантируемой характеристики территории, которая не может быть измерена напрямую. Примером такой характеристики может служить гидротермический коэффициент Селянинова, который показывает уровень влагообеспеченности территории. Гидротермический коэффициент рассчитывается на основе данных об осадках за период с температурой воздуха выше +5 °C (PCP) и суточных температурах за этот же период (TSUM) по формуле $GTK = \frac{10 \times PCP}{TSUM}$ [41, 42].
2. Построение карт пригодности территорий по совокупности лимитирующих факторов среды. Примером здесь может служить создание обобщенных карт экологической пригодности территорий для произрастания того или иного вида растений, которые используются в сельском хозяйстве при интродукции видов в новые для них регионы. Такие карты строятся на основе карт экологической пригодности по суммам активных температур за некоторые периоды времени и карт экологической пригодности по уровню влагообеспеченности [43].

Операция растровой алгебры принимает в качестве своих аргументов задаваемую пользователем формулу расчета и набор растровых слоев, значения точек которых будут подставляться в формулу. Все исходные растры должны иметь одинаковое разрешение $n \times m$ точек. Для того, чтобы полученный в результате выполнения операции растровый слой содержал осмысленные данные, исходные растры также должны обладать одинаковыми картографической проекцией и ограничивающим прямоугольником. Полученный в результате выполнения операции слой будет иметь те же разрешение, картографическую проекцию и ограничивающий прямоугольник, что и исходные слои.

Пусть L – это множество растровых слоев, имеющих одинаковые разрешение, картографическую проекцию и ограничивающий прямоугольник. Тогда, операция растровой алгебры может быть представлена функцией:

$$f \colon L^n \to L, n \in \mathbb{N}$$

Причем, каждая конкретная функция определяется пользователем при «заказе» операции растровой алгебры. Примером конкретной функции может служить уже упомянутая формула расчета гидротермического коэффициента: $GTK = \frac{10 \times PCP}{TSUM}$.

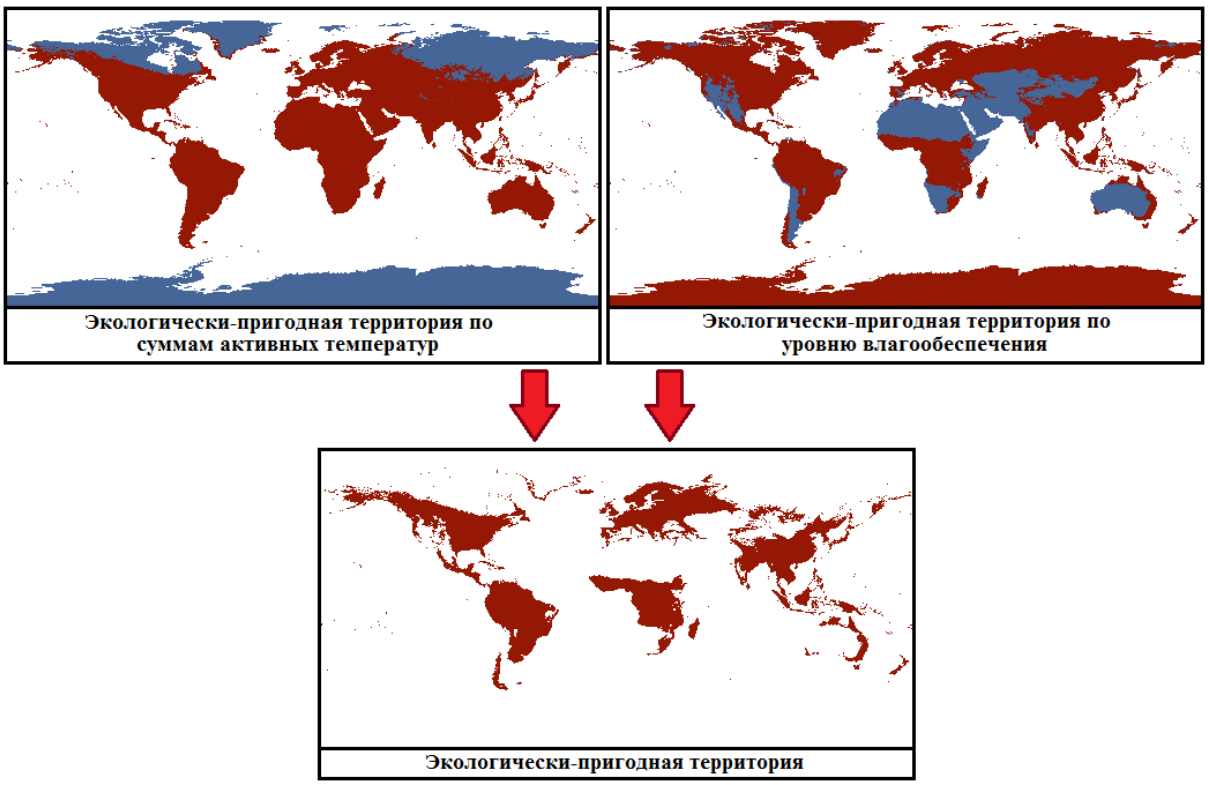

Рис. 5: Использование растровой алгебры для выделения экологически-пригодных для проживания вида территорий. Итоговая ЭПТ получается путем пересечения ЭПТ по отдельным факторам.

Поскольку операция растровой алгебры рассматривает каждую точку растрового слоя независимо от остальных, все элементы мозаики растра могут обрабатываться параллельно. Другими словами, выполнение растровой алгебры над «большими» слоями может быть значительно ускорено путем разбиения обрабатываемых растров на отдельные небольшие фрагменты, каждый из которых обрабатывает отдельным узлом вычислительной сети.

Стоит также отметить, что время, необходимое на выполнение растровой алгебры зависит не только от разрешения растров и их количества, но также и от сложности заданной пользователем функции. При прочих равных, пользовательские функции, использующие только арифметические действия, будут вычисляться значительно быстрее, чем те, что используют показательные или тригонометрические функции.

Выполнение операций растровой алгебры и растровой реклассификации может быть заметно ускорено путем использования графических процессоров, имеющих программируемые шейдерные блоки. Графические процессора обладают высокой эффективностью выполнения операций растрового анализа поскольку они способны обрабатывать «пиксели» растровых слоев целыми блоками, а не по одной точке за раз, как центральный процессор [44, 45].

5.2.4. Визуализация данных

Визуализация пространственных данных является очень важной вспомогательной операцией, существенно облегчающей анализ данных человеком. Без визуализации пространственных данных крайне сложно дать ответы на такие важные вопросы эколого-географического анализа как «Какие земли наилучшим образом подходят для выращивания X?» или «На каких территориях в ближайшее время существует высокая вероятность возникновения эпидемиологических очагов Y?»

Поскольку визуализация пространственных данных в растровых форматах также является операцией, рассматривающей каждую точку растрового слоя независимо от остальных, выполнение данной операции может быть во многих случаях значительно ускорено благодаря грамотному распределению элементов мозаики по узлам сети.

Это связано со следующим свойством растровой мозаики: в пользовательском окне с разрешением, не превосходящим разрешение отдельного элемента мозаики, в каждый момент времени могут находиться территории, расположенные на 1, 2 или 4 элементах мозаики соответствующего уровня пирамиды. При этом, эти элементы мозаики будут располагаться на различных узлах сети, и, в силу этого, могут быть обработаны параллельно без необходимости в пересылке данных между узлами вычислительной сети.

Также, стоит отметить, что каждый элемент мозаики на уровне пирамиды N соответствует четырем смежным элементам мозаики на уровне пирамиды $N + 1$. Эти элементы также будут располагаться на различных узлах вычислительной сети и могут быть обработаны параллельно. В зависимости от распределения нагрузки на сеть и положения пользовательского окна, использование элементов мозаики, расположенных на более низком уровне пирамиды, может быть предпочтительным с точки зрения уменьшения времени отклика системы.

Таким образом, в ситуации, когда разрешение пользовательского окна меньше или равно разрешению отдельного элемента растровой мозаики, вне зависимости от того, какую именно часть слоя просматривает пользователь, всегда имеется возможность визуализировать эту часть слоя за время, не превышающее время, необходимое для визуализацию одного элемента мозаики.

Если же разрешение пользовательского окна превосходит разрешение отдельного элемента мозаики, то такое окно может рассматриваться как несколько смежных пользовательских окон меньшего размера. Таким образом, сколько бы элементов мозаики в действительности не находилось в пользовательском окне, визуализация данных этой части слоя всегда может быть выполнена за время, не превышающее время, необходимое на визуализацию N элементов мозаики, где N – минимальное число элементов растровой мозаики текущего уровня пирамиды, которое необходимо для заполнения пользовательского окна.

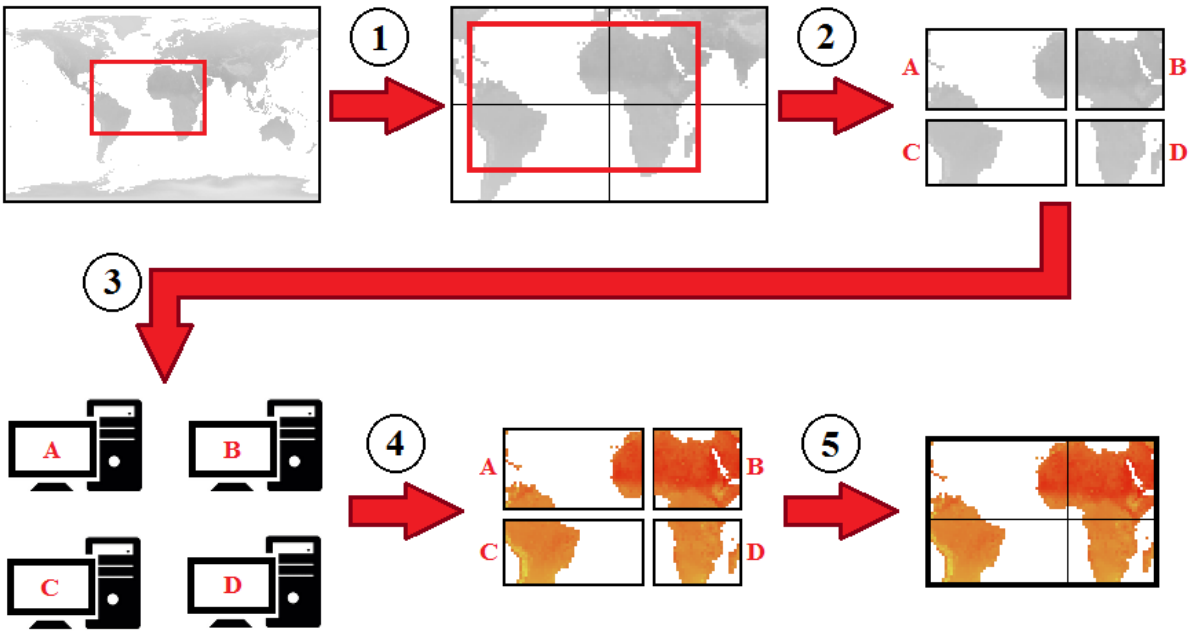

Рис. 6: Распределенная визуализация пространственных данных в вычислительной сети:
1. Определение положения пользовательского окна на просматриваемом слое.
2. Выявление элементов растровой мозаики, попадающих в пользовательское окно.
3. Передача узлам вычислительной сети задания на визуализацию элементов мозаики.
4. Получение результатов визуализации элементов мозаики от узлов вычислительной сети.
5. Формирование окончательного изображения и отправка его пользователю системы.

Помимо непосредственно визуализации данных, важной также является доставка полученных изображений до конечного пользователя системы. Для распределенных многопользовательских ГИС, работающих в добровольных вычислительных сетях, одним из наилучших решений для передачи через локальную сеть географически привязанных изображений видится использование протокола WMS [46]:

- Протокол WMS позволяет ретранслировать данные между серверами. Таким образом. является возможным организовать распределенную ГИС в которой визуализация данных будет проводиться на тех же вычислительных узлах, где эти данные хранятся. Пользователю будут передаваться только полученные в результате визуализации графические изображения, в результате чего нагрузка на каналы связи будет минимальной.
- Практически все современные ГИС поддерживают работу с протоколом WMS. Пользователи создаваемой распределенной ГИС смогут взаимодействовать с системой, используя наиболее привычный и удобным им интерфейс.
- При наличии подключения предлагаемого прототипа ГИС к сети Интернет, пользователи смогут получать из него данные, находясь в любой точке мира. Это является крайне полезным свойством для тех групп пользователей, которые часто работают «в поле».

Для того, чтобы создаваемая распределенная ГИС поддерживала протокол WMS необходимо, чтобы узлы вычислительной сети могли отвечать на следующие запросы со стороны пользователей системы [46]:

- Запрос **GetCapabilities** – ответом на данный запрос является информация о параметрах запущенного на сервере WMS сервиса, а также список доступных слоев с базовой информацией о них.
- Запрос **GetMap** – ответом на данные запрос является изображение, которое является визуализацией слоя или его отдельной области. Данный запрос имеет ряд стандартных параметров, которые определяют: какую область какого слоя и в какой картографической проекции необходимо визуализировать; разрешение и формат запрашиваемого изображения; а также стилизацию, которая должна быть использована при визуализации данных.

Кроме приведенных выше запросов, поддержка которых должна осуществляться в обязательном порядке, также существуют запросы, поддержка которых является опциональной для WMS-сервисов [46]:

- Запрос **GetFeatureInfo** – данный запрос позволяет получать информацию о значениях в некоторой точке рассматриваемого слоя. Данный запрос имеет ряд стандартных параметров, которые определяют: в какой области какого слоя располагается интересующая пользователя точка; формат данных для передачи пользователю запрашиваемой информации; максимальное число полей данных, которое может быть включено в отчет; а также формат для предоставления отчета об ошибке в случае ее возникновения.
- Запрос **DescribeLayer** – ответом на данный запрос является описание слоя, включая настройки его визуализации по-умолчанию.
- Запрос **GetLegendGraphic** – ответом на этот запрос является изображение, содержащее список условных обозначений либо иные пояснения к визуализации рассматриваемого слоя.

Перечисленные выше обязательные и опциональные запросы на которые должен отвечать WMS-сервер не создают препятствий для нормального функционирования модулей распределенной географической информационной системы, ответственных за хранение и обработку пространственных данных. Таким образом, включение в состав разрабатываемой ГИС WMS-сервера дает возможность повысить комфорт пользователей при работе с системой при нулевом или минимальном негативном влиянии на функционирование системы.

5.3. Географическая информационная система

Далее в этом разделе будет рассмотрено внутреннее устройство разрабатываемого прототипа распределенной ГИС для добровольных и гибридных вычислительных сетей. В системе могут быть выделены следующие ключевые компоненты:

- Распределенная файловая система, решающая такие задачи, как распределение сохраняемых данных по узлам вычислительной сети; их репликация с целью повышения доступности; а также поддержание согласованности при внесении изменений в хранимые данные. Кроме этого, распределенная файловая система предоставляет информацию о физическом расположении тех или иных данных, позволяя повысить производительность системы за счет минимизации числа пересылок данных между узлами в ходе выполнения вычислений.
- Подсистема мониторинга состояния узлов вычислительной сети. В задачи этого компонента системы входит мониторинг состояния узлов системы. Если узел системы становится недоступным, то выполнявшиеся на нем задачи должны быть перезапущены на других узлах вычислительной сети. Данная подсистема определяет, когда именно возникает необходимость в перезапуске задач.
- Планировщик задач. Задачами этого модуля является распределение заданий на обработку данных между узлами вычислительной сети, а также перезапуск заданий, выполнявшихся на узле, если тот по каким-либо причинам становится недоступным.
- Вычислительный модуль, с помощью которого выполняются пользовательские операции над данными. Именно этот модуль реализует операции растровой реклассификации и растровой алгебры, необходимые осуществления эколого-географического анализа пространственных данных в растровых форматах. Без данного модуля создаваемый прототип ГИС является неспособным выполнять свою основную функцию: анализ пространственных данных.

Указанные компоненты создаваемого прототипа распределенной ГИС будут рассмотрены более детально далее в этом разделе.

5.3.1. Распределенная файловая система

Обеспечение долговечности, доступности и согласованности хранимых «больших данных» является одной из важнейших задач создаваемой ГИС. И решение этой задачи возлагается на входящую в состав программного комплекса распределенную файловую систему. Именно от нее в конечном счете зависит, как будут распределены

данные по узлам вычислительной сети, что во многих случаях влияет на скорость выполнения многих операций над этими данными: если хранимые данные хорошо сгруппированы по узлам сети, то достигается максимальная параллелизация в их обработке, а необходимость в затратной операции пересылки данных между узлами, наоборот, практически отсутствует.

Также, для того, чтобы создаваемый прототип мог рационально распределять задания на обработку данных между узлами вычислительной сети, используемая распределенная файловая система также должна предоставлять информацию о том, где именно располагаются те или иные фрагменты данных. Это условие означает, что используемая файловая система должна относиться к классу файловых систем с запоминанием физического адреса данных (Location-aware File Systems).

Однако, поскольку разрабатываемая географическая информационная система предназначается для работы в добровольных и гибридных вычислительных сетях, существует ряд дополнительных факторов, которые необходимо учитывать при выборе файловой системы для данной ГИС:

- В добровольных вычислительных сетях узлы могут часто и на продолжительное время становиться недоступными. По результатам экперимента, проведенного в вычислительном центре Калифорнийского университета в Сан Диего, узлы в добровольных вычислительных сетях могут быть недоступны 40% времени [22]. Это делает многие распределенные файловые системы неэффективными для использования в добровольных вычислительных сетях, поскольку они начинают «метаться» в попытках поддержать заданное количество репликаций данных в вычислительной сети.

- Включение в добровольную вычислительную сеть группы выделенных узлов делает представление фактора репликации при помощи единственного числа неадекватным, поскольку сеть теперь состоит из двух различных классов узлов. Как следствие, в используемой файловой системе должна иметься возможность задания факторов репликации файла отдельно для выделенных и добровольных узлов вычислительной сети.

- В добровольных вычислительных сетях пользователи могут использовать свои компьютеры для решения каких-либо сторонних задач. Т.е., узлы добровольной вычислительной сети должны быть полностью функциональными машинами и вне этой сети. Проще говоря, данные на узлах добровольных вычислительных сетях могут быть манипулированы при помощи каких-либо средств, отличных от распределенной файловой системы ГИС, и это не должно оказывать влияния на файловую систему ГИС.

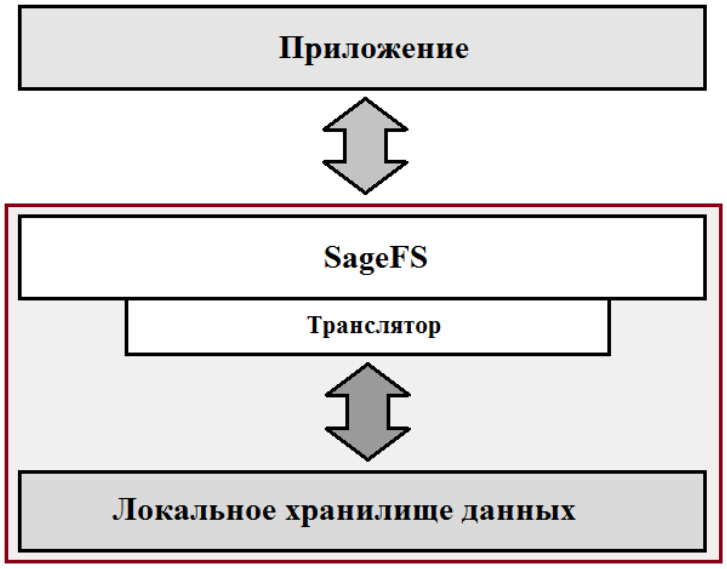

Рис. 7: Архитектура SageFS.

- Также, добровольная вычислительная сеть с высокой степенью вероятности является гетерогенной, как в плане аппаратного обеспечения, так и в плане программного обеспечения. Это касается и локальных файловых систем узлов вычислительной сети. Распределенная файловая система в составе создаваемой ГИС должна уметь работать с этими локальными файловыми системами.
- Для того, чтобы данные систематично записывались на определенные узлы, приложения, работающие с файловой системой в составе создаваемой ГИС, должны иметь возможность указать, на какие именно узлы вычислительной сети они хотят поместить тот или иной фрагмент данных.

Кроме того, используемая в создаваемой ГИС распределенная файловая система должна обладать минимальными накладными расходами. Также, поскольку число возлагаемых на нее задач также невелико: создание, модификация, репликация и удаление файлов, а также предоставление планировщику заданий информации о физическом расположении данных на узлах вычислительной сети, хорошей идеей может быть использование не полноценной распределенной файловой системы, а компактной клиентской библиотеки, реализующей указанные функции.

В качестве такой легковесной библиотеки была выбрана библиотека `SageFS`. Эта библиотека создает общее API для работы с множеством разных бэкэндов, а также интегрирует их вместе, представляя их пользовательским приложениям как единую файловую систему. `SageFS` использует трансляторы для манипуляции с файлами на бэкэндах, жестко привязывая каждый экземпляр транслятора к какому-то из бэкэндов. Библиотека `SageFS` позволяет пользователям явно задавать, с помощью каких трансляторов читать или сохранять файл. А, поскольку каждый транслятор

явно связан с конкретным бэкэндом, это позволяет явным образом определять, с каких узлов вычислительной сети файл будет читаться или на какие узлы файл будет записываться.

Также, группируя трансляторы вместе, можно задавать репликацию файла, в том числе и для случая добровольных вычислительных сетей, где число необходимых репликаций файла в системе задается не одним значением, а парой значений $\{d, v\}$, где d – число репликаций данных на выделенных узлах, а v – на добровольных.

Наконец, благодаря реализации доступа к данным через трансляторы, в `SageFS` динамическое или удаление бэкэндов является возможным. В силу этого `SageFS` может работать в добровольных вычислительных сетях, где узлы могут часто и на продолжительное время становиться недоступными. Группировка трансляторов как основа репликации данных также означает, что `SageFS` не будет «метаться» в попытках поддержать заданное количество репликаций данных в ситуации, когда какой-либо из узлов вычислительной сети вдруг стал недоступным.

Альтернативой SageFS может служить модификация HDFS, адаптированная для работы в добровольных вычислительных сетях. Однако, поскольку в стандартной реализации HDFS репликации данных распределяются по узлам вычислительной сети неконтролируемым образом [48, 49], использование HDFS не позволит добиться значительного ускорения при выполнении операций, требующих чтения данных из более чем одного файла, поскольку будет существовать постоянная необходимость в пересылке данных между узлами вычислительной сети. Существующие решения по изменению политики HDFS распределения репликаций блоков данных по узлам вычислительной сети – например Hadoop++ или CoHadoop – также не являются оптимальными.

5.3.2. Мониторинг состояния узлов вычислительной сети

Как уже было сказано ранее, в добровольных вычислительных сетях узлы могут часто становиться недоступными: компьютер выключен, или находится в режиме ожидания, или же, если система настроена таким образом, нагрузка от локальных процессов превысила некоторое критическое значение. И для того, чтобы система не посылала задания на узлы, которые не могут их в настоящий момент выполнить, для системы, работающей в добровольных вычислительных сетях, мониторинг состояния узлов вычислительной сети является важной задачей.

Обычно, состояние узла сети наблюдается при помощи специальных сообщений, «пульса». Если от узла в течении определенного интервала времени не поступает сигналов пульса, то такой узел считается «мертвым», и предназначающиеся ему задания перераспределяются между другими узлами сети.

Такой подход хорошо работает, когда все узлы вычислительной сети являются выделенными и имеют уровень доступности близкий к 100%. В добровольных же сетях, уровень доступности узлов может быть 60% или даже ниже [22]. Узлы часто «выпадают» из сети: компьютер выключен, или находится в режиме ожидания, или нагрузка от локальных процессов превысила некоторое критическое значение.

Максимальный временной интервал между сообщения пульса, при котором узел сети не будет помечен как «мертвый» определяется значением `NodeExpiryInerval`. И если в добровольной сети он задан слишком маленьким, то узлы сети будут часто менять свои состояния, заставляя систему выполнять лишние действия по отмене и перезапуску заданий. С другой стороны, если интервал `NodeExpiryInerval` задан слишком длинным, то система может ошибочно полагать что какой-то из узлов все еще является «живым» и отправлять на него задания на обработку данных. Это также приводит к затратам ресурсов системы впустую и таймаутам при попытках доступа пользователями системы к узлам.

Соответственно, требуется механизм, который позволит определять, испытывает ли тот или иной узел вычислительной сети кратковременную потерю доступности в связи с, например, возросшей нагрузкой от локальных процессов, или же этот узел «умер» и надо перераспределить его задания между другими узлами системы.

Оригинальное решение данной задачи было предложено разработчиками системы MOON [22]. Их идея заключается в добавлении дополнительного состояния узла: «бездействующий» (hibernating). Узел, пребывающий в этом состоянии по-прежнему входит в состав вычислительной сети, однако не может в настоящее время выполнять задания по каким-то причинам. Бездействующие узлы не получают новых заданий, но уже отправленные им пока не передаются другим узлам, поскольку существует надежда, что данный узел скоро «оживет» и продолжит выполнять свои задания.

В системе MOON, если пульс узла не приходит в течение `NodeHibernateInterval`, который значительно короче `NodeExpiryInerval`, то такой узел системы помечается как бездействующий, и новые задания на него не отправляются. Если же пульс узла не прослушивается в течение `NodeExpiryInerval`, то такой узел помечается системой как «мертвый», а все отправленные на него задания перезапускаются на других узлах вычислительной сети.

Использование такого подхода позволяет уменьшить число перераспределений заданий в добровольной вычислительной сети при частых кратковременных потерях доступности узлами сети.

Данный подход к определению состояний узлов добровольной вычислительной сети представляется подходящим для использования в создаваемой распределенной географической информационной системе.

5.3.3. Планирование заданий

Планировщик заданий является одним из важнейших компонентов распределенной вычислительной системы. Именно он определяет, выполнение каких заданий будет поручено тем или иным узлам. Как следствие, именно от планировщика заданий во многом зависит насколько быстро и эффективно вычислительная система может решать поставленные перед ней пользователями задачи.

Планировщику важно учитывать расположение данных на узлах вычислительной сети при планировании заданий. И, при наличии возможности, задания должны планироваться на узлах, которые уже содержат необходимые для выполнения этого задания данные. Если же это по каким-то причинам невозможно, то для выполнения задания должен быть выбран узел, максимально близко расположенный к месту хранения необходимых данных. Это позволит минимизировать пересылки данных между узлами сети, таким образом заметно повышая производительность системы. В некоторых ситуациях, ускорение вычислений благодаря отсутствию пересылок данных между узлами сети может достигать 12% или более [20].

Планировщик заданий также должен наблюдать за состоянием выполняемых в данный момент заданий и, если узел на котором это задание выполняется перестает подавать сигналы пульса, то такое задание должно быть перезапущено на других узлах вычислительной сети, желательно также имеющих необходимые данные в локальном доступе.

Также должен учитываться тот факт, что обработка пространственных данных является процессом, у которого в пределах отдельного элемента растровой мозаики отсутствуют контрольные точки. Если при обработке элемента мозаики узел сети становится недоступным, то при перезапуске задачи на другом узле вычислительной сети необходимо начинать вычисления по этому элементу мозаики с самого начала. Это означает, что задания, имеющие повышенный приоритет, а также задания на обработку больших и сверх-больших «неделимых» блоков данных, следует запускать на выделенных узлах вычислительной сети, где шанс, что узел станет недоступным во время выполнения задания, значительно ниже.

С другой стороны, обработка различных фрагментов растровой мозаики будет занимать примерно одинаковое время на вычислительных узлах с примерно равной производительностью. Также, при выполнении операций растрового анализа слой обрабатывается целиком, что дает возможность распределить задания на обработку фрагментов растровой мозаики более-менее поровну между узлами вычислительной сети. Таким образом, ситуации, когда какое-то из отдельных заданий выполняется значительно дольше остальных при штатном функционировании вычислительной сети практически исключены. В силу этого планировщик заданий распределенной ГИС, работающей в добровольных вычислительных сетях, может обойтись и без

дополнительных правил репликации заданий на «финишной прямой» выполнения пользовательской задачи, которые необходимы для эффективной работы системы общего назначения в добровольных сетях.

Таким образом, планировщик заданий для распределенной ГИС, работающей в добровольной вычислительной сети, в ходе своей работы должен отталкиваться от следующих соображений:

- При выборе узла для выполнения задания на обработку фрагментов растровой мозаики предпочтение должно отдаваться узлам вычислительной сети, которые уже хранят копию данных этого фрагмента мозаики. Это позволит избежать затрат на пересылку данных, повышая тем самым производительность системы.
- Задачи пользователей с высоким приоритетом, а также задания на обработку растров, представленных крупными фрагментами, должны распределяться на выделенные узлы вычислительной системы. «Рядовые» задачи, при наличии возможности, должны распределяться на добровольные узлы.
- Задания, в особенности требующие много времени для своего выполнения, не должны «убиваться» как только выполняющему их узлу присваивается статус «бездействующего» – существует шанс, что этот узел станет снова доступным в ближайшее время и продолжит выполнение задания. Перезапуск задач должен осуществляться только когда выполнявший их узел помечается системой как «мертвый». При обработке пространственных «больших данных» такой подход позволит минимизировать излишние затраты ресурсов, вызванные поспешными отменой и перезапуском заданий.

5.3.4. Вычислительный модуль

Вычислительный модуль создаваемой географической информационной системы реализует в себе важные операции по обработке и анализу растровых данных. Без данного модуля предлагаемый прототип ГИС не будет способен выполнять операции эколого-географического анализа, и, как следствие, будет бесполезен.

В вычислительном модуле для прототипа ГИС реализованы следующие операции по обработке и анализу растровых данных:

- Вспомогательный метод, задачей которого является подготовка растрового слоя к использованию в системе. Данный метод разбивает исходный растровый слой на мозаику с задаваемым пользователем разрешением элемента, выполняет построение пирамиды, а также распределяет полученные фрагменты данных по узлам вычислительной сети.

- Метод, отвечающий за визуализацию растровых данных. Этот метод принимает в качестве своих аргументов фрагмент растровой мозаики и ограничивающий прямоугольник пользовательского окна, и возвращает изображение в формате PNG с разрешением 1 пиксель = 1 визуализированная точка растра. Алгоритм определения фрагментов мозаики, подлежащих визуализации, представлен в **приложении 1**; алгоритм раскрашивания растра приведен в **приложении 2**.
- Метод, реализующий операцию растровой реклассификации. Данный метод принимает в качестве своих аргументов элемент растровой мозаики, параметры реклассификации в виде массива чисел, и имя, под которым будут сохраняться полученные результаты. Алгоритм растровой реклассификации представлен в **приложении 3**.
- Метод, реализующий операцию растровой алгебры. Данный метод принимает в качестве своих аргументов формулу, по которой будут проводиться вычисления, а также имя, под которым будут сохраняться полученные результаты. Имена используемых в растровой алгебры слоев выделяются из формулы. Алгоритм растровой алгебры представлен в **приложении 4**.

Указанные 4 метода являются минимальным набором методов, необходимых для проведения эколого-географического анализа в географической информационной системе [20, 42, 43].

6. Полученные результаты

Для проверки жизнеспособности предложенной концепции распределенной ГИС был выполнен прототип, реализующий минимальный набор функций, необходимых для работы системы. Далее, этот прототип был подвергнут ряду тестов с целью определения того, насколько эффективно он может решать задачи географической информационной системы. В качестве тестовых данных были использованы мировые растровые слои в картографической проекции EPSG:4326 с размерами исходных файлов 100 и 400 Мб, что соответствует разрешению слоев в 25 и 100 МПикс.

Тестируемая система была развернута на вычислительной сети из 8 узлов, 4 из которых играли роль выделенных узлов, а оставшиеся 4 были добровольными. После построения мозаики и пирамиды, размер файлов слоев увеличился до 131.25 и 531.25 Мб, из которых на каждый узел приходилось по 37.5 и 137.5 Мб соответственно.

Далее, были проведены три группы тестов по 20 тестов в каждой:

- Тесты на скорость визуализации данных: системе требовалось визуализировать случайным образом расположенное пользовательское окно. Разрешение данного окна было задано равным разрешению одного элемента растровой мозаики.
- Тесты на скорость выполнения растровой реклассификации: прототипу ГИС требовалось выполнить операцию растровой реклассификации над случайно выбранным слоем заданного размера.
- Тесты на скорость выполнения растровой алгебры: от системы требовалось выполнить операцию растровой алгебры над парами слоев одинакового размера.

Полученные результаты затем были сравнены с результатами аналогичных тестов для географической информационной системы, работающей на 1 узле.

Для исключения влияния разницы в реализации алгоритмов обработки данных на получаемые результаты, и локальная и распределенная ГИС используют один и тот же вычислительный модуль. Также, во всех тестах вычисления проводились на 1 ядре центрального процессора с тактовой частотой 3.34 ГГц. Все системы могли использовать до 4 Гб оперативной памяти.

6.1. Визуализация данных

В ходе первой группы тестов, от системы требовалось выполнить визуализацию случайным образом расположенного пользовательского окна, разрешение которого было задано равным разрешению одного элемента растровой мозаики. Это окно могло быть расположено на любом допустимом уровне приближения и покрывало 1, 2 или 4 элемента растровой мозаики соответствующего уровня пирамиды.

При тестировании замерялось время от прихода на управляющий узел системы команды на визуализацию данных до момента завершения передачи полученного изображения пользователю. Поскольку размеры элементов растровой мозаики для слоев разрешением 25 и 100 МПикс равны – больший слой просто разбит на большее число элементов мозаики – единственным фактором, который может значительно влиять на время визуализации, является число элементов мозаики, покрываемых пользовательским окном.

В ходе тестирования были получены следующие результаты:

Рис. 8: Результаты тестов на скорость визуализации пользовательского окна.

Как можно видеть из Рис. 8, распределенная система работает немного медленнее в ситуациях, когда требуется визуализировать данные только 1 элемента растровой мозаики, но показывает заметно лучший результат, когда требуется визуализация большего числа элементов мозаики. Это связано с тем, что распределенная система может обрабатывать несколько элементов растровой мозаики параллельно, но ей необходимо время на планирование заданий, а также на передачу данных между управляющим и рабочими узлами.

6.2. Растровая реклассификация

В ходе данной группы тестов, от ГИС требовалось выполнить операцию растровой реклассификации случайным образом выбранного растрового слоя из коллекции. При реклассификации, значения точек выбранного растра могли быть разбиты на $2 - 5$ классов. Подобная растровая реклассификация хорошо симулирует реальные операции, выполняемые в ходе выделения экологически-пригодных территорий в ходе выполнения эколого-географического анализа [43].

При тестировании замерялось время от прихода на управляющий узел системы команды на выполнение операции до завершения выполнения операции для системы на одном узле и до завершения репликации данных для распределенной системы. Поскольку время выполнения операции растровой реклассификации значительно зависит от размеров реклассифицируемого слоя, полученные результаты для слоев с исходными размерами 100 и 400 Мб будут приведены по-отдельности.

В ходе тестирования были получены следующие результаты:

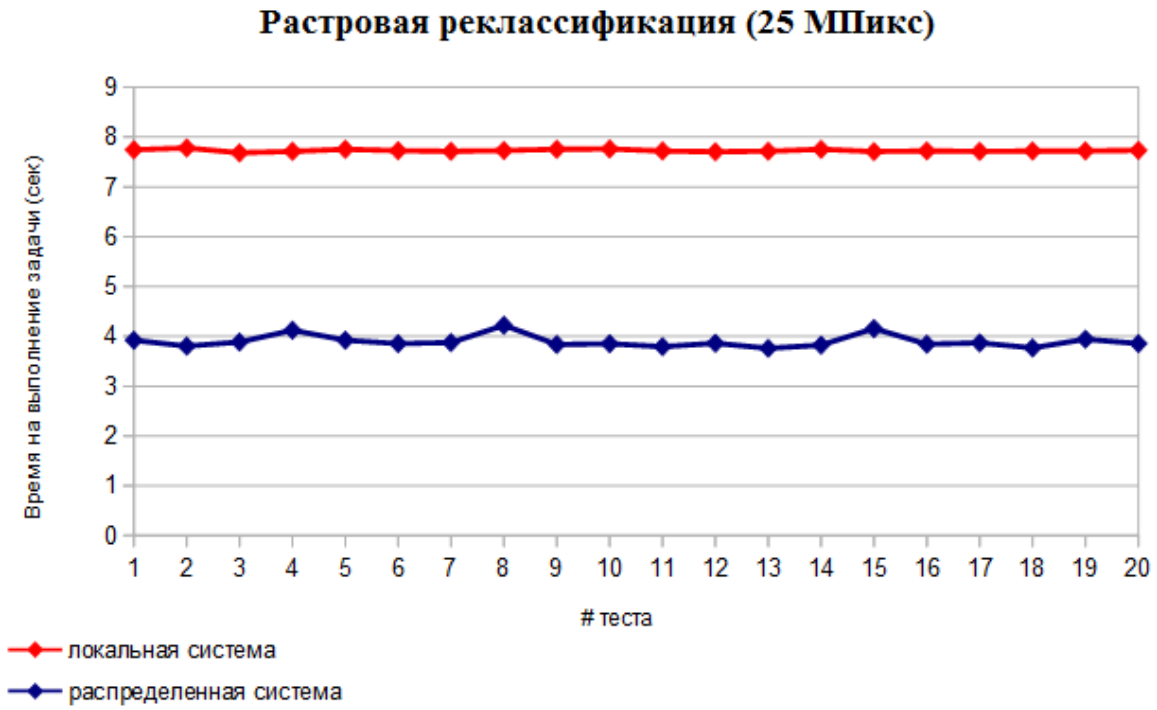

Рис. 9: Результаты тестов на скорость выполнения растровой реклассификации слоя 25 МПикс.

При растровой реклассификации слоя разрешением 25 МПикс локальная система показала среднее время 7.728 секунды на обработку 21 элемента растровой мозаики. Распределенная система при выполнении растровой реклассификации тех же слоев показала среднее время 3.898 секунды на обработку 6 элементов растровой мозаики на каждом из 4 вычислительных узлов и репликацию полученных результатов.

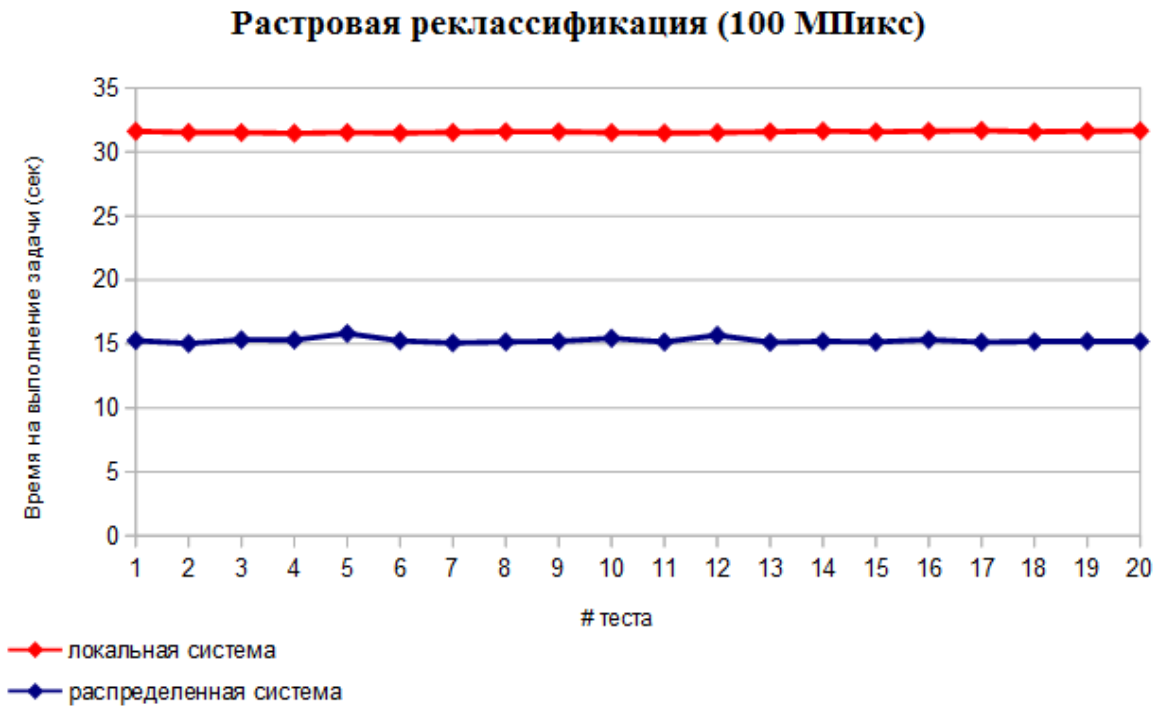

Рис. 10: Результаты тестов на скорость выполнения растровой реклассификации слоя 100 МПикс.

При реклассификации слоя разрешением 100 МПикс локальная система показала среднее время 31.57 секунды на обработку 85 элементов растровой мозаики. При выполнении растровой реклассификации этих же слоев, распределенная система показала среднее время 15.27 секунды на обработку 22 элементов растровой мозаики на каждом из 4 вычислительных узлов с репликацией полученных результатов.

6.3. Растровая алгебра

В ходе данной группы тестов, ГИС должна была выполнить растровую алгебру над парой случайно выбранных растровых слоев одного размера. К значениям точек растров применялись следующие арифметические операции: сложение, вычитание, умножение, деление. Подобная растровая алгебра достаточно хорошо симулирует уровень операций, которые используются при выполнении эколого-географического анализа [42, 43].

При тестировании замерялось время от прихода на управляющий узел системы команды на выполнение операции до завершения выполнения операции для системы на одном узле и до завершения репликации данных для распределенной системы. Поскольку время выполнения операции растровой алгебры значительно зависит от размеров участвующих в операции растровых слоев, полученные результаты для слоев с исходными размерами 100 и 400 Мб будут приведены по-отдельности.

В ходе тестирования были получены следующие результаты:

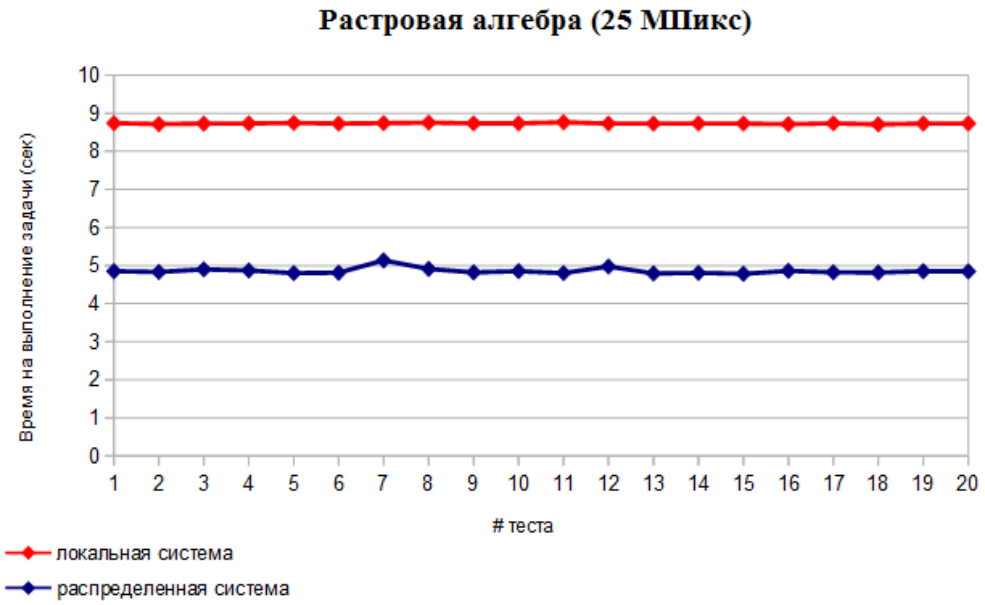

Рис. 11: Результаты тестов на скорость выполнения растровой алгебры над слоями 25 МПикс.

В ходе выполнении растровой алгебры над растровыми слоями с разрешением 25 МПикс, локальная система показала среднее время 8.733 секунды на обработку 21 пары элементов растровой мозаики. Распределенная система при выполнении аналогичных действий показала среднее время 4.858 секунды на обработку 6 пар элементов растровой мозаики на каждом из 4 вычислительных узлов и репликацию полученных результатов.

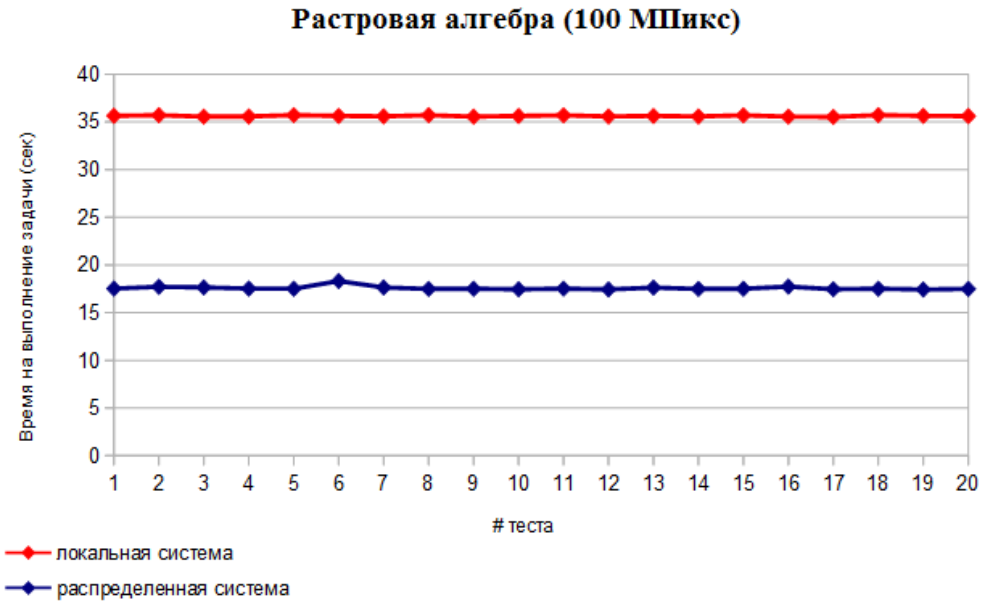

Рис. 12: Результаты тестов на скорость выполнения растровой алгебры над слоями 100 МПикс.

В ходе выполнении растровой алгебры над растровыми слоями с разрешением 100 МПикс, локальная система показала среднее время 35.63 секунды на обработку 85 пар элементов растровой мозаики. Распределенная же система при выполнении аналогичных действий показала среднее время 17.58 секунды на обработку 22 пар элементов растровой мозаики на каждом из 4 вычислительных узлов и репликацию полученных результатов.

6.4. Анализ полученных результатов

Полученные в ходе тестирования результаты указываются на то, что тестируемый прототип распределенной географической информационной системы в большинстве случаев выполняет поставленные пользователями задачи быстрее, чем аналогичная по функционалу ГИС, работающая на только одном вычислительном узле.

- При визуализации пользовательского окна распределенная система показывает более-менее стабильное время выполнения задания на уровне примерно $0.7 - 0.8$ секунды, в то время как результаты локальной системы сильно зависят от числа фрагментов растровой мозаики, которые требуется обработать, и варьируются между 0.5 и 2 секундами.

  В большинстве случаев, распределенная система справляется с визуализацией пользовательского окна быстрее локальной системы. Единственным случаем, когда это не так, является ситуация, когда в пределах пользовательского окна находятся территории, представленные на одном фрагменте растровой мозаики. В этом случае, время, затраченное на планирование заданий распределенной системой, не может быть отыграно за счет параллельной обработкой данных. Такая ситуация, однако, случается довольно редко, а, значит, распределенная система в среднем справляется с задачей визуализации пользовательского окна быстрее, чем ее локальный аналог.

- При выполнении операций растровой реклассификации и растровой алгебры, распределенная система, состоящая из 4 вычислительных узлов, справлялась с пользовательскими заданиями в среднем в 1.968 раза быстрее, чем локальная система, несмотря на затраты времени на планирование заданий и репликацию полученных результатов.

  Стоит отметить, что для тестов на скорость выполнения растровой алгебры были использованы крайне простые формулы, использующие данные только 2 различных слоев. При использовании более сложных формул или большего числа слоев, выполнение расчетов будет требовать больше времени. Затраты времени на планирование заданий и репликацию данных при этом останутся

прежними. Таким образом, чем более сложные действия выполняются в ходе растровой алгебры, тем более значительным будет выигрыш распределенной ГИС по сравнению с локальной.

Также, поскольку время на планирование заданий не зависит от разрешения элементов растровой мозаики, при обработке бóльших фрагментов растровой мозаики планирование заданий будет занимать меньшую долю времени, таким образом увеличивая выигрыш, который может быть получен от использования распределенной ГИС.

Наконец, увеличение числа вычислительные узлов, параллельно работающих над обработкой «больших данных», позволяет повысить производительность распределенной системы в целом.

Ну и нельзя не отметить, что в распределенной географической информационной системе пользователь имеет возможность работать с бóльшим объемом данных, чем может вместить жесткий диск его персонального компьютера, что также является плюсом распределенных ГИС по сравнению с настольными.

Итого, по результатам проведенного тестирования, предлагаемый прототип ГИС продемонстрировал возможность эффективно хранить пространственные данные в добровольной вычислительной сети с их репликацией для обеспечения доступности. При выполнении операций растрового анализа и визуализации пользовательского окна, созданный прототип ГИС показал в среднем лучшее время, чем настольная ГИС, использующая идентичный вычислительный модуль. Таким образом было показано, что использование ГИС, работающих в добровольных или гибридных вычислительных сетях, является целесообразным и оправданным при работе с пространственными «большими данными».

Преимущества от использования распределенных систем становятся наиболее очевидными при работе с растровыми слоями с разрешением в несколько ГПикс. Для слоев с таким разрешением, даже небольшое увеличение скорости выполнения операций эколого-географического анализа может вылиться в ощутимую экономию времени [1].

[1] Мировой слой в проекции EPSG:4326 с пространственным разрешением 110 м экватора на 1 пиксель будет иметь разрешение 360000 × 180000 пикселей. Для выполнения реклассификации данного слоя использованным в данной работе вычислительным модулем потребуется примерно 4 с половиной часа времени при использовании ресурсов одного компьютера.

7. Заключение

Развитие технологий в современном мире позволяет человеку иметь доступ ко все бóльшим объемам данных из самых различных областей жизнедеятельности, и эколого-географический анализ не является исключением. Ежедневно и ежечасно цифровые каталоги информации об экологии, погоде, климате и распространении биологических объектов уточняются и дополняются усилиями сотен и тысяч людей и автоматических систем, ведущих наблюдение с поверхности Земли, с самолетов и атмосферных зондов, а также из космоса.

И рост объемов собранной информации постоянно ускоряется: растет как число источников данных, так и детализация самих данных. Пространственные данные в современном мире без тени сомнения являются «большими данными». А так как пространственные данные в настоящее время активно используются в различных областях научной и повседневной деятельности человека, возникает необходимость в надежных и производительных вычислительных системах для хранения и обработки пространственных данных.

Возможным вариантом таких систем являются распределенные географические информационные системы, работающие в добровольных вычислительных сетях. Эти ГИС сочетают в себе высокую производительность, репликацию данных, и большие объемы доступной долговременной памяти, присущие вычислительным кластерам, с возможностями независимого использования входящих в состав вычислительной сети персональных компьютеров.

Использование таких распределенных географических информационных систем позволяет выполнять эколого-географический анализ пространственных «больших данных» за разумное время даже без использования высокопроизводительного оборудования. Благодаря этому, даже небольшие лаборатории получат возможность проводить виртуальные биологические и экологические эксперименты, используя для этого карты с высоким пространственным разрешением, или же заниматься прогнозированием распространения опасных биологических объектов, раскрывая секреты окружающего нас мира и делая его безопаснее для жизни.

Благодарности

Автор данной работы выражает свою глубокую и искреннюю признательность коллективу факультета Прикладной Математики - Процессов Управления СПбГУ и, в особенности, своим научным руководителям, к.ф.-м.н. Сергееву Сергею Львовичу и Севрюкову Сергею Юрьевичу, за время и усилия вложенные ими в его обучение, а также за оказанную ими поддержку при планировании и написании данной работы.

Автор также выражает свою глубокую признательность коллективу лаборатории Моделирования и Диагностики Геосистем института Наук о Земле СПбГУ и лично к.с.-х.н. Афонину Александру Николавичу за интродукцию в мир географических информационных систем и за предоставленные материалы и тестовые данные, использованные при создании и тестировании представленного в данной работе прототипа распределенной ГИС для добровольных вычислительных сетей.

Большое Вам всем спасибо!

Литература

[1] Грант «*Эколого-географическое исследование распространения агробактерий и растений, имеющих в геноме последовательности ДНК агробактериального происхождения*», ИАС: 0.37.526.2013.

[2] Hu H., Wen Y., Chua T.-S., Li X. Towards scalable systems for big data analytics: a technology tutorial // IEEE Xplore Digital Library. 2014. `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6842585` (Дата обращения: 04.04.2016)

[3] J. Gantz, D. Reinsel. Extracting value from chaos // Proc. IDC iView. 2011. P. 1–12.

[4] Snijders C., Matzat U., Reips U. D. 'Big Data': Big gaps of knowledge in the field of Internet. // International Journal of Internet Science 7 (2012). P. 1–5.

[5] D. Laney 3D Data Management: Controlling Data Volume, Velocity, and Variety // META Group Inc. File 949. 2001.

[6] P. Zikopoulos, C. Eaton Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data // New York, NY, USA: McGraw-Hill, 2011.

[7] E. Meijer The world according to LINQ // Commun. ACM. vol 54. No 10. P. 45–51. 2011.

[8] J. Manyika et al. Big data: The Next Frontier for Innovation, Competition, and Productivity // McKinsey Global Institute paper. P. 1–137. 2011.

[9] Targio H. et al. "Big data"on cloud computing: Review and open research issues. // Information Systems 47 (2015). P. 98–115.

[10] De Mauro A., Greco M., Grimaldi M. What is big data? A consensual definition and a review of key research topics. // AIP Conference Proceedings (2015). P. 97–104.

[11] M. Cooper, P. Mell Tackling Big Data // 2012. `http://csrc.nist.gov/groups/SMA/forum/documents/june2012presentations/fcsm_june2012_cooper_mell.pdf` (дата обращения 04.04.2016)

[12] Processing: What to record? [Электронный ресурс] // `http://home.cern/about/computing/processing-what-record` (дата обращения: 05.04.2016)

[13] HDFS Architecture Guide [Электронный ресурс] // `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html` (дата обращения: 05.04.2016)

[14] Tredger S. SageFS: the location aware wide area distributed filesystem [Электронный ресурс]: URL: https://dspace.library.uvic.ca:8443/bitstream/handle/1828/5824/Tredger_Stephen_MSc_2014.pdf?sequence=3&isAllowed=y (дата обращения: 07.04.2016)

[15] NoSQL Databases Explained [Электронный ресурс] // https://www.mongodb.com/nosql-explained (дата обращения: 05.04.2016)

[16] I.Katsov, D. Kirkdorffer NoSQL Data Modeling Techniques [Электронный ресурс] // https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/ (дата обращения: 05.04.2016)

[17] E. Nightingale, J. Elson, J. Fan, O. Hofmann, J. Howell, Y. Suzue, Flat Datacenter Storage // 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '12)

[18] V. Sandeep, K. Yerlanki, Flat Datacenter Storage [Электронный ресурс] // http://www.ece.eng.wayne.edu/~sjiang/ECE7650-winter-15/topic4A-S.pdf (дата обращения: 11.04.2016)

[19] C. Modi, Flat Datacenter Storage [Электронный ресурс] // http://www.ece.eng.wayne.edu/~sjiang/ECE7650-winter-15/topic4B-S.pdf (дата обращения: 11.04.2016)

[20] Соловьев П.А. Использование сетевых файловых систем с запоминанием физического адреса данных для ускорения обработки больших объемов пространственных данных // Процессы управления и устойчивость. 2015. Т. 2(18). № 1. С. 509-514.

[21] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters [Электронный ресурс] // http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf (дата обращения: 15.04.2016)

[22] H. Lin et al, MOON: MapReduce On Opportunistic eNvironments [Электронный ресурс] // http://eprints.cs.vt.edu/archive/00001089/01/moon.pdf (дата обращения: 15.04.2016)

[23] C. Yang, C. Yen, C. Tan, S. R. Madden, Osprey: Implementing MapReduce-Style Fault Tolerance in a Shared-Nothing Distributed Database [Электронный ресурс] // http://db.csail.mit.edu/pubs/OspreyDB.pdf (дата обращения: 15.04.2016)

[24] Z. Ma, L. Gu, The limitation of MapReduce: A probing case and a lightweight solution // Proceeding of the 1st Intl. Conf. on Cloud Computing, GRIDs, and Virtualization, 2010.

[25] Z. Ma, K. Hong, L. Gu, MapReduce-style Computation in Distributed Virtual Memory [Электронный ресурс] // http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.115&rep=rep1&type=pdf (дата обращения: 15.04.2016)

[26] M. A. Kozuch et al, Tashi: Location-aware Cluster Management [Электронный ресурс] // http://www.pdl.cmu.edu/PDL-FTP/Storage/tashi-acdc2009.pdf (дата обращения: 16.04.2016)

[27] сайт директивы INSPIRE Европейского Сообщества // http://inspire.jrc.ec.europa.eu/

[28] S. Tschirner, A. Scherp, S. Staab, Semantic access to INSPIRE: how to publish and query advanced GML data [Электронный ресурс] // http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.662.5258&rep=rep1&type=pdf (дата обращения: 26.04.2016)

[29] M. Zhang el al, TerraFly GeoCloud: An Online Spatial Data Analysis and Visualization System [Электронный ресурс] // http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.705.6789&rep=rep1&type=pdf (дата обращения: 26.04.2016)

[30] W. Kuhn, Geospatial Semantics: Why, of What, and How [Электронный ресурс] // http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.504.1688&rep=rep1&type=pdf (дата обращения: 26.04.2016)

[31] A. Crow, S. Aerni, H. Farinas, R. Radhakrishnan, G. Muralidhar, Data Science How To: Massively Parallel, In-Database Image Processing [Электронный ресурс] // https://blog.pivotal.io/big-data-pivotal/features/data-science-how-to-massively-parallel-in-database-image-processing-part-1 (дата обращения: 28.04.2016)

[32] Y. Yan, L. Huang, Large-Scale Image Processing Research Cloud [Электронный ресурс] // https://www.thinkmind.org/download.php?articleid=cloud_computing_2014_4_20_20069 (дата обращения: 28.04.2016)

[33] Oracle Spatial GeoRaster Developer's Guide [Электронный ресурс]: URL: http://docs.oracle.com/cd/B28359_01/appdev.111/b28398/geor_intro.htm (дата обращения: 09.03.2015)

[34] Xu Hong, Mangtani P. Managing imagery and raster data using mosaic datasets // ESRI International User Conference. Technical Workshop. 2012.

[35] Описание возможностей файлового формата BigTiff [Электронный ресурс] // http://bigtiff.org/ (дата обращения: 10.05.2016)

[36] The BigTIFF File Format Proposal [Электронный ресурс] // http://www.awaresystems.be/imaging/tiff/bigtiff.html (дата обращения: 10.05.2016)

[37] Y. Pessach, Distributed Storage: Concepts, Algorithms and Implementations // CreateSpace Independent Publishing Platform; 1 edition. 2013

[38] P. Bailis, A. Ghodsi, Eventual Consistency Today: Limitations, Extensions, and Beyond // ACM Queue, Vol. 11, Issue 3. 2013

[39] Запорожченко А. В., Картографические проекции и методика и выбора для создания карт различных типов // Ногинск: Панорама, 1991-2007. 148 с.

[40] Официальный сайт библиотеки GDAL [Электронный ресурс] // http://www.gdal.org/ (дата обращения: 10.05.2016)

[41] Афонин А. Н., Грин С. Л., Дзюбенко Н. И., Фролов А. Н. (ред.) Агроэкологический атлас России и сопредельных стран: экономически значимые растения, их вредители, болезни и сорные растения. 2008. 1 электрон. опт. диск (DVD).

[42] Афонин А. Н., Ли Ю. С., Эколого-географический подход на базе географических информационных технологий в изучении экологии и распространения биологических объектов // BioGIS Journal. 2011, N 1.

[43] Carpenter G., Gillison A. N., Winter J., DOMAIN: a flexible modelling procedure for mapping potential distributions of plants and animals. // Biodiversity and Conservation 2, 1993. C. 667-680.

[44] J. Fung, S. Mann, Using Multiple Graphics Cards as a General Purpose Parallel Computer: Applications to Computer Vision // Proceedings of the 17th International Conference on Pattern Recognition (ICPR2004), vol. 1, pp. 805–808

[45] D. Göddeke, Fast and Accurate Finite-Element Multigrid Solvers for PDE Simulations on GPU Clusters. [Электронный ресурс] // http://d-nb.info/100545535X/34 (дата обращения: 15.05.2016)

[46] Документация по протоколу WMS [Электронный ресурс] // http://www.opengeospatial.org/standards/wms (дата обращения: 15.05.2016)

[47] Репозиторий клиентской библиотеки SageFS [Электронный ресурс] // https://github.com/stredger/sagefs (дата обращения: 15.05.2016)

[48] M. Y. Eltabakh, Y. Tian, F. Özcab, R. Gemulla, A. Krettek, J. McPherson, CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop [Электронный ресурс] // http://researcher.watson.ibm.com/researcher/files/us-ytian/colocation.pdf (дата обращения: 19.05.2016)

[49] M. G. Ferriera, Replication and Data Placement in Distributed Key-Value Stores [Электронный ресурс] // http://www.gsd.inesc-id.pt/~ler/reports/manuelferreira-midterm.pdf (дата обращения: 19.05.2016)

Приложение 1: Определение положения пользовательского окна

При визуализации пространственных данных в географических информационных системах одной из задач, требующих решения, является определение тех элементов растровой мозаики, которые покрываются пользовательским окном.

В представленном в данной работе прототипе распределенной системы, данная задача решается при помощи следующего алгоритма:

Используемые переменные:

Request - Полученное задание на обработку данных.

LayerID - Идентификатор рассматриваемого слоя.

LayerData - Подробное описание рассматриваемого слоя.

LayerBBOX - Ограничивающий прямоугольник слоя.

WindowBBOX - Ограничивающий прямоугольник пользовательского окна.

LayerSpan - Объект с двумя полями, в которых хранится размер огрaчивающего прямоугольника слоя по осям X и Y.

WindowSpan - Объект с двумя полями, в которых хранится размер огрaчивающего прямоугольника окна пользователя по осям X и Y.

Zoom - Уровень пирамиды с которого брать данные.

Coeff - Отношения размеров элемента мозаики на уровне пирамиды 0 к размеру элемента мозаики на уровне пирамиды Zoom.

TileSpanX - Ширина ограничивающего прямоугольника элемента мозаики на уровне пирамиды Zoom.

TileSpanY - Высота ограничивающего прямоугольника элемента мозаики на уровне пирамиды Zoom.

Xmin, Ymin - "Координаты"элемента мозаики на который приходится левый верхний угол пользовательского окна.

Xmax, Ymax - "Координаты"элемента мозаики на который приходится правый нижний угол пользовательского окна.

TheTile - Объект для временного хранения какого-либо из элементов мозаики.

Tiles - Список элементов мозаики, которые необходимо обработать для выполнения поставленного пользователем задания.

```
LayerID = Request -> Layer;
LayerData = GETLAYERDATA (LayerID);
LayerBBOX = LayerData -> BBOX;
WindowBBOX = Request -> BBOX;
LayerSpan -> X = LayerBBOX -> xmax - LayerBBOX -> xmin;
LayerSpan -> Y = LayerBBOX -> ymax - LayerBBOX -> ymin;
WindowSpan -> X = WindowBBOX -> xmax - WindowBBOX -> xmin;
WindowSpan -> Y = WindowBBOX -> ymax - WindowBBOX -> ymin;
Zoom = 0; Coeff = 1;
WHILE WindowSpan -> X * Coeff < LayerSpan -> X AND
      WindowSpan -> Y * Coeff < LayerSpan -> Y DO
    Zoom = Zoom + 1;
    Coeff = Coeff * 2;
END WHILE
TileSpanX = LayerSpan -> X / Coeff;
TileSpanY = LayerSpan -> Y / Coeff;
```

Теперь определим элемент мозаики, на который приходится верхний левый угол пользовательского окна. Элементы растровой мозаики нумеруются начиная с 0 с запада на восток и с севера на юг.

```
DispX = WindowBBOX -> xmin - LayerBBOX -> xmin;
DispY = LayerBBOX -> ymax - WindowBBOX -> ymax;
Xmin = FLOOR (DispX / TileSpanX);
IF Xmin < 0 THEN
    Xmin = 0;
END IF
Ymin = FLOOR (DispY / TileSpanY);
IF Ymin < 0 THEN
    Ymin = 0;
END IF
```

Теперь определим элемент мозаики, на который приходится нижний правый угол пользовательского окна.

```
DispX = WindowBBOX -> xmax - LayerBBOX -> xmin;
DispY = LayerBBOX -> ymax - WindowBBOX -> ymin;
Xmax = FLOOR (DispX / TileSpanX);
IF Xmax < 0 THEN
    Xmax = 0;
END IF
```

```
Ymax = FLOOR (DispY / TileSpanY);
IF Ymax < 0 THEN
    Ymax = 0;
END IF
```

Теперь поместим все элементы мозаики, частично или полностью покрываемые пользовательским окном, в массив для дальнейшей обработки

```
FOR i = Xmin; i <= Xmax; i = i + 1 DO
    FOR j = Ymin; j <= Ymax; j = j + 1 DO
        TheTile = NEW TILE(LayerID, i, j, Zoom);
        Tiles -> PUSH (TheTile);
    END FOR
END FOR
```

Предложенный алгоритм предполагает использование структуры `REQUEST` для хранения информации о заказанном пользователем задании. Структура типа `REQUEST` включает в себя поля с информацией о типе заказанной операции и ее параметрах; о слое к которому данная операция должна быть применена; а также о размерах пользовательского окна (актуально только для операции визуализации данных).

Также, предложенный алгоритм предполагает использование структуры `TILE` для хранения информации о конкретном элементе растровой мозаики. Структура типа `TILE` содержит в себе поля с информацией о том, какому слою принадлежит данный элемент мозаики, и где именно он располагается.

Приложение 2: Алгоритм визуализации растровых данных

Визуализация пространственных данных, в особенности представленных растрами, является важной вспомогательной операцией, в значительной степени облегчающей анализ данных человеком. Визуализация растровых данных помогает дать ответы на такие важные вопросы эколого-географического анализа как «Какие территории наилучшим образом подходят для выращивания X?» или «В каких областях сейчас существует вероятность возникновения эпидемиологических очагов Y?»

В представленном в данной работе прототипе распределенной географической информационной системы для добровольных вычислительных сетей визуализация растров выполняется при помощи следующего алгоритма:

Используемые переменные:

Pixels - Данные, которые требуется визуализировать.

Width - Ширина генерируемого изображения.

Height - Высота генерируемого изображения.

Max - Максимальное значение визуализируемого растра.

Min - Минимальное значение визуализируемого растра.

Img - Объект типа BITMAP IMAGE.

Palette - Класс, хранящий описание палитры, которая используется для раскрашивания растра.

Color - Переменная для хранения значения цвета.

Stream - Поток, в который сохраняется генерируемое изображение.

Format - Файловый формат, в котором будет сохранено полученное изображение.

Temp - Вспомогательная переменная.

```
Min = Bytes -> Min();
Max = Bytes -> Max();
Img = NEW BITMAP(Width, Height);
FOR i = 0; i < Height; i = i + 1 DO
    FOR j = 0; j < Width; j = j + 1 DO
        Temp = (Max - Pixels[i * Width + j]) / (Max - Min);
```

```
        Color = Palette -> GETCOLOR(Temp);
        Img -> SETPIXEL(j, i, Color);
    END FOR
END FOR

Img -> SAVE(Stream, Format);
Img -> DISPOSE();
```

Далее, полученное изображение может быть, например, извлечено из потока, закодировано в Base64 и отправлено пользователю.

Представленный алгоритм предполагает использование класса `PaletteEntity` (переменная `Palette`), который хранит описание палитры цветов, выбранное для раскрашивания растра. Этот класс реализует метод `GETCOLOR`, который принимает в качестве своего единственного аргумента положение значения рассматриваемой точки растра на отрезке $[0, 1]$, где 0 соответствует минимальному значению на визуализируемом фрагменте растра, а 1 – максимальному. Реализуется этот метод следующим образом:

Используемые переменные:

IndexA - Индекс ключевого цвета, предшествующего «позиции» раскрашиваемой точки растра.

IndexB - Индекс ключевого цвета, следующего за «позицией» раскрашиваемой точки растра.

ColorA - Ключевой цвет с индексом IndexA.

ColorB - Ключевой цвет с индексом IndexS.

PosA - «Позиция» ключевого цвета с индексом IndexA.

PosB - «Позиция» ключевого цвета с индексом IndexB.

Pos - «Позиция» раскрашиваемой точки растра.

Colors - Отсортированный по позициям массив ключевых цветов палитры.

TheColor - Цвет, присвоенный рассматриваемой точке растра.

R, G, B - Компоненты цвета рассматриваемой точки растра.

Temp - Вспомогательная переменная.

```
IndexA = 0;
IndexB = -1;
FOR i = 0; i < COUNT(Colors); i = i + 1 DO
    Temp = Colors[i] -> GETPOSITION();
    IF Pos = Temp THEN
```

```
        RETURN Colors[i] -> GETCOLOR();
    ELSE IF Temp < Pos THEN
        IndexA = i;
    ELSE IF Temp > Pos AND IndexB = -1 THEN
        IndexB = i;
    END IF
END FOR

ColorA = Colors[IndexA] -> GETCOLOR();
ColorB = Colors[IndexB] -> GETCOLOR();
PosA = Colors[IndexA] -> GETPOSITION();
PosB = Colors[IndexB] -> GetPosition();
Temp = (PosB - Pos) / (PosB - PosA);
R = FLOOR (ColorA -> R * Temp + ColorB -> R * (1 - Temp));
G = FLOOR (ColorA -> G * Temp + ColorB -> G * (1 - Temp));
B = FLOOR (ColorA -> B * Temp + ColorB -> B * (1 - Temp));
TheColor = NEW COLOR(R, G, B);
RETURN TheColor;
```

Представленная реализация метод `GETCOLOR` класса `PaletteEntity` предполагает использование классов `COLOR` для хранения цвета (модель RGB), а также списка объектов типа `PaletteColor`, представляющих собой ключевые цвета используемой палитры. Объект типа `PaletteColor` содержит в себе поле цвета (типа `COLOR`) и поле, которое определяет позицию данного ключевого цвета палитры на отрезке $[0, 1]$, где 0 соответствует цвету минимального значения на визуализируемом фрагменте растра, а 1 – цвету максимального значения на визуализируемом фрагменте растра.

Методы `GETCOLOR` и `GETPOSITION` возвращают цвет и позицию ключевого цвета палитры соответственно.

Приложение 3: Алгоритм растровой реклассификации

Растровая реклассификация является одной из основных операций растрового анализа и используется для определения областей, значения точек растра в которых лежит в пределах задаваемых пользователем границ.

Операция растровой реклассификации принимает в качестве своих аргументов один растровый слой и таблицу, в соответствие с которой будут замещаться значения на данном растре. Результатом выполнения растровой реклассификации является новый слой с теми же разрешением, проекцией и топографической привязкой, что и исходный слой.

Растровая реклассификация реализуется программно следующим образом:

Используемые переменные:

MultiStripped - Флаг, который указывает, на то каким способом значения точек растра хранятся в файле. Принимает значение «Ложь», если данные хранятся единым массивом, и значение «Истина», если данные разбиты на полосы.

InFile - Файл с исходными данными.

OutFile - Файл, в который записывается результат.

Offsets - Массив в котором хранятся точки начала полос.

StripSizes - Массив размеров полос данных в файле слоя. Используется только если MultiStripped имеет значение «Истина».

StripsCount - Число полос данных в файле.

PointsLeft - Число еще необработанных точек в файле.

Bytes - Сколько байт памяти занимает одно значение.

BlockSize - Число точек растра, обрабатываемых за одно чтение из файла.

ABlockSize - Число точек в текущем считанном блоке.

Points - Массив, в который читаются точки из файла.

Params - Таблица замещения значений. Представлена массивом длины $3 * N$. Индексы $3 * i$ содержат нижние границы «классов», $3 * i + 1$ - верхние границы, а $3 * i + 2$ - присваиваемые значения.

ParamsCount - Число «классов» в таблице замещений.

```
IF MultiStripped IS FALSE THEN
    FILESEEK (InFile, Offsets[0]);
    WHILE PointsLeft > 0 DO
        IF PointsLeft < BlockSize THEN
            Points = FILEREAD (InFile, PointsLeft * Bytes);
            ABlockSize = PointsLeft;
            PointsLeft = 0;
        ELSE
            PointsLeft = PointsLeft - BlockSize;
            ABlockSize = BlockSize;
            Points = FILEREAD (InFile, BlockSize * Bytes);
        END IF

        i = 0;
        WHILE i <= ABlockSize DO
            FOR j = 0; j < ParamsCount; j = j + 1 DO
                IF Points[i] > Params[j * 3] AND
                   Points[i] < Params[j * 3 + 1] THEN
                       Points[i] = Params[j * 3 + 2];
                       BREAK;
                END IF
            END FOR
            i = i + 1;
        END WHILE
        FILEWRITE (OutFile, Points);
        EMPTY (Points);
    END WHILE
ELSE
    FOR s = 0; s <= StripsCount; s = s + 1 DO
        FILESEEK (InFile, Offsets[s]);
        ABlockSize = StripSizes[s] / Bytes;

        i = 0;
        WHILE i <= ABlockSize DO
            FOR j = 0; j < ParamsCount; j = j + 1 DO
                IF Points[i] > Params[j * 3] AND
                   Points[i] < Params[j * 3 + 1] THEN
                       Points[i] = Params[j * 3 + 2];
                       BREAK;
                END IF
            END FOR
            i = i + 1;
        END WHILE
        FILEWRITE (OutFile, Points);
        EMPTY (Points);
    END FOR
END IF
```

Приложение 4: Алгоритм растровой алгебры

Растровая алгебра является одной из основных операций растрового анализа и активно используется при проведении эколого-географическом анализе для решения следующих задач:

- Генерация растровых слоев, содержащих информацию о значениях какой-либо квантируемой характеристики территории, которая не может быть измерена напрямую.
- Построение карт пригодности территорий по совокупности лимитирующих факторов среды.

Операция растровой алгебры принимает в качестве своих аргументов задаваемую пользователем формулу расчета и непустое множество растровых слоев, значения точек которых будут подставляться в формулу. Все исходные растры должны иметь одинаковое разрешение $n \times m$ точек. Полученный в результате выполнения операции слой будет иметь те же разрешение, что и исходные слои.

Растровая алгебры реализуется программно следующим образом:

Используемые переденные:

FileCount - Число обрабатываемых файлов.

MultiStripped - Флаг, который указывает, на то каким способом значения точек растра хранятся в файле. Принимает значение «Ложь», если данные хранятся единым массивом, и значение «Истина», если данные разбиты на полосы.

InFiles - Массив файлов с исходными данными.

OutFile - Файл, в который записывается результат.

Offsets - Двухмерный массив в котором хранятся точки начала полос в файлах.

StripSizes - Двухмерный массив размеров полос в файлах.

PointsLeft - Число еще необработанных точек в растрах.

Bytes - Массив размеров значений в байтах.

BlockSize - Число точек, обрабатываемых за одно чтение из файла.

ABlockSize - Массив числа еще не обработанных точек в текущем считанном блоке.

CStrips - Массив индексов обрабатываемых сейчас полос.

Points - Двухмерных массив, в который считываются значения из файлов.

CPoints - Массив индексов текущих точек в Points.

OutPoints - Массив, в который записываются полученные в ходе обработки данных результаты.

SExpression - Строка с текстом функции, которая будет применена к растровым слоям.

Expression - Объект класса математического выражения, который строится по задаваемой пользователем системы функции (переменная SExpression).

Values - Массив значений, которые будут подставляться в математическое выражение (переменная Expression).

TotalPoints - Общее число точек в растрах.

ВНИМАНИЕ! В данный псеводкод проверка идентичности разрешений обрабатываемых растровых слоев НЕ включена.

```
Expression = NEW EXPRESSION();
Expression -> INITIALIZE (SExpression);

WHILE PointsLeft > 0 DO
    FOR i = 0; i < FileCount; i = i + 1 DO
        IF ABlockSize[i] = 0 THEN
            IF NOT IS_EMPTY (OutPoints) THEN
                FILEWRITE (OutFile, OutPoints);
                EMPTY (OutPoints);
            END IF
            CPoints[i] = 0;
            IF MultiStripped[i] IS TRUE THEN
                FILESEEK (InFiles[i], Offsets[i][CStrips[i]]);
                PointsToRead = StripSizes[i][CStrips[i]];
                BytesToRead = PointsToRead * Bytes[i];
                Points[i] = FILEREAD (InFiles[i], BytesToRead);
                CStrips[i] = CStrips[i] + 1;
            ELSE
                PointsDone = TotalPoints - PointsLeft;
                Offset = Offstes[i][0] + PointsDone * Bytes[i];
                FILESEEK (InFiles[i], Offset);
                IF PointsLeft > BlockSize THEN
                    ABlockSize[i] = BlockSize;
                ELSE
                    ABlockSize[i] = PointsLeft;
                END IF
```

```
                BytesToRead = ABlockSize[i] * Bytes[i];
                Points[i] = FILEREAD (InFiles[i], BytesToRead);
            END IF
        END IF
    END FOR

    FOR i = 0; i < FileCount; i = i +1 DO
        Values[i] = Points[i][CPoints[i]];
        CPoints[i] = CPoints[i] + 1;
    END FOR
    Result = Expression -> EVALUATE (Values);
    OutPoints -> PUSH (Result);
    PointsLeft = PointsLeft - 1;
END WHILE
IF NOT IS_EMPTY (OutPoints) THEN
    FILEWRITE (OutFile, OutPoints);
    EMPTY (OutPoints);
END IF
```

Предложенный алгоритм растровой алгебры подразумевает использование некого класса `EXPRESSION`, для которого реализованы следующие два метода:

1. `INITIALIZE`: принимает в качестве своего единственного аргумента текстовую строку, содержащую текст функции, которая должна быть применена к точкам обрабатываемых растровых слоев. Данный метод не возвращает значение, но формирует внутри экземпляра класса `EXPRESSION` дерево разбора выражения, которое может быть вычислено с помощью, например, стековой машины.
2. `EVALUATE`: принимает в качестве своего единственного аргумента массив чисел, которые должны быть подставлены в дерево разбора выражения, полученное при вызове метода `INITIALIZE`, на место переменных, соответствующих точкам слоев. После того, как значения были подставлены, производится вычисление данного дерева. Полученный результат возвращается в основную программу.

Например:

Для примера рассмотрим формулу расчета гидротермического коэффициента Селянинова. Пусть `L1` и `L2` являются переменными, которым будут присваиваться числовые значения из аргумента метода `EVALUATE`.

```
Expression -> INITIALIZE ( "L1 * 10 / L2" );
```

Вызов метода `INITIALIZE` формирует в объекте `Expression` дерево разбора выражения (в Обратной Польской Нотации): $L1 \quad 10 \quad * \quad L2 \quad /$.

```
Expression -> EVALUATE( {2, 5} );
```

При вызове метода `EVALUATE` переменным `L1` и `L2` в дереве разбора выражения присваиваются значения 2 и 5 соответственно. Итоговое дерево рабора выражения будет иметь вид: 2 10 * 5 /. При вычислении этого дерева разбора будет получено значение 4, которое и будет возвращено методом `EVALUATE`.